UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

GRADO INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FINAL DE GRADO

MYLEARNINGMENTOR

DESARROLLO DE APLICACIÓN PARA SISTEMA OPERATIVO ANDROID

Autor: Ricardo García Pericuesta

Tutor: Carlos Alario Hoyos

Directora: Iria Manuela Estévez Ayres

Resumen

Los MOOCs, cursos en línea masivos y abiertos, ofrecen la posibilidad de acceder a contenidos y recursos educativos de calidad a cualquier persona, independientemente de su ubicación geográfica o situación económica. Estos cursos son de una gran variedad temática y son ofrecidos por universidades de todo el mundo, a través de plataformas como Coursera, edX, Miríadax, encontrándose entre ellas las universidades más prestigiosas actualmente. Sin embargo, estos cursos, que cada vez están más presentes en la vida de las personas, tienen una tasa de abandono muy elevada, típicamente superior al 90%. Los MOOCs son una excelente opción para adquirir conocimientos y por ello se pretende motivar y ayudar a las personas a que aprovechen y finalicen este tipo de cursos.

Con este objetivo se plantea la creación de una herramienta que dé soporte a los estudiantes que participan en los MOOCs. Para facilitar la adopción de esta herramienta y facilitar al usuario el acceso a la misma, se decide desarrollar una aplicación móvil, a la cual las personas puedan acceder e instalar en sus propios dispositivos de forma sencilla. Aprovechando las facilidades para el desarrollo y distribución de la aplicación que ofrece el sistema operativo Android, se decide implementar una aplicación para este sistema operativo.

Para el desarrollo de dicha aplicación se seguirá una arquitectura cliente-servidor y se desarrollarán ambas partes. Como resultado, se obtendrá una aplicación lista para ser utilizada por cualquier estudiante de un MOOC. Las funciones principales que se ofrecen al usuario le permitirán llevar al día los diferentes MOOCs en los que se haya matriculado. El objetivo final de la aplicación será ayudar a los estudiantes a finalizar sus estudios. Antes de finalizar el trabajo se realizarán diferentes pruebas para validar que la aplicación realmente cumple con los requisitos establecidos. Por último, se expondrán las conclusiones que se han ido extrayendo en la realización de la aplicación.

Contenido

Resumen ..... **II**

Contenido ..... **III**

Índice de ilustraciones ..... **V**

Índice de tablas ..... **VI**

Acrónimos ..... **VII**

Capítulo 1: Introducción **1**

1.1 Contexto **1**

1.2 Motivación **2**

1.3 Objetivos **3**

1.4 Estructura de la memoria **3**

CAPITULO 2: MyLearningMentor **5**

2.1 MyLearningMentor **5**

2.1.1 Introducción 5

2.1.2 Arquitectura 6

2.1.3 Interfaz 7

2.2 Estudio y elección de las tecnologías **8**

2.2.1 Sistemas operativos móviles inteligentes. 8

2.2.2 Ejemplo de sistemas operativos móviles 9

2.2.3 Arquitecturas de sistemas distribuidos. 13

2.2.4 Sistemas de gestión de bases de datos 15

2.2.5 Sistemas de gestión documental 16

2.2.6 Lenguajes de parte del servidor 16

2.2.7 Comunicación entre cliente servidor. 17

2.3 Conclusiones **18**

Capítulo 3: Refinamiento del diseño e implementación **19**

3.1 Refinamiento del diseño **19**

3.1.1 Requisitos 19

3.1.2 Arquitectura 20

3.1.3 Funcionalidad 23

3.1.3 Diseño de las bases de datos 24

3.2 Implementación **28**

3.2.1 Servidor 29

3.2.2 Conexión servidor-sistemas de almacenamiento de datos 31

3.2.3 Cliente 32

3.2.4 Diagrama de clases 34

3.2.5 Diagramas de secuencia 41

3.3 Conclusiones **44**

Capítulo 4: Validación **45**
4.1 Validación de requisitos funcionales **45**
4.2 Pruebas de carga **53**
4.3 Seguridad de la aplicación **54**
4.4 Conclusiones **55**
Capítulo 5: Conclusiones y líneas futuras **56**
5.1 Conclusiones **56**
5.2 Líneas futuras. **57**
Abstract **58**
Extended abstract **59**
Anexo A: Planificación **71**
Anexo B: Presupuesto **72**
Anexo C: Marco legal **74**
Anexo D: Actividades del cliente **75**
Bibliografía **82**

Índice de ilustraciones

Ilustración 1: Distribución de los MOOCs en Europa 1
Ilustración 2: Distribución de los MOOCs por temática en Europa 2
Ilustración 3: Arquitectura de MyLearningMentor 7
Ilustración 4: Capturas de pantallas de MyLearningMentor 8
Ilustración 5: Evolución de PCs, iPhone & Android smartphones y tablets 9
Ilustración 6: Cuotas de los sistemas operativos sobre nuevos smartphones 10
Ilustración 7: Ejemplo de arquitectura cliente-servidor 14
Ilustración 8: Pruebas de velocidad. MongoDB-MySQL 21
Ilustración 9: Documento de la colección TaskCourse 27
Ilustración 10: Documento de la colección Course-user-task 28
Ilustración 11: Esquema de funcionalidades de MyLearningMentor 29
Ilustración 12: Actividad MyPlanner 33
Ilustración 13: Actividad SeeTask 33
Ilustración 14: Actividad SearchCourse 33
Ilustración 15: Clase Httppostaux 34
Ilustración 16: Clases del bloque de funcionalidades Tratamiento de usuarios (I) y Consejos .. 35
Ilustración 17: Clases del bloque de funcionalidades Tratamiento de usuarios (II) 36
Ilustración 18: Clases para los bloques de funcionalidades Planificador Adaptativo, Consejos y Realimentación 37
Ilustración 19: Clases para el bloque de funcionalidades Cursos 38
Ilustración 20: Clases del bloque de requisitos tareas 39
Ilustración 21: Clases del bloque de requisitos tareas (Editar y Añadir tareas) 40
Ilustración 22: Diagrama de secuencia del planificador adaptativo 41
Ilustración 23: Diagrama de secuencia para listar los cursos matriculados por un alumno 42
Ilustración 24: Diagrama de secuencia AddCour 43
Ilustración 25: Capturas de pantalla prueba Registro en la aplicación 46
Ilustración 26: Capturas de pantalla prueba Actualización del perfil académico 47
Ilustración 27: Capturas de pantalla prueba Añadir un nuevo curso 49
Ilustración 28: Capturas de pantalla prueba Matriculación en un curso ya existente 50
Ilustración 29: Capturas de pantalla con fallo de servidor 54
Ilustración 30: Diagrama de Gantt 71

Índice de tablas

Tabla 1: Requisitos de la aplicación 5
Tabla 2: Comparativa Android e iOS 13
Tabla 3: Campos de la base de datos “Profiles” 25
Tabla 4: Campos de la base de datos "Tips” 25
Tabla 5: Campos de la base de datos “Mooc_Data” 26
Tabla 6: Campos de la base de datos “URL_COURSE” 26
Tabla 7: Tecnologías utilizadas 28
Tabla 8: Resumen de los archivos del servidor 30
Tabla 9: Conexión entre servidor y sistemas de almacenamiento de datos 32
Tabla 10: Relación entre cada actividad y los recursos de servidor que requiere 33
Tabla 11: Validación de las funcionalidades de la aplicación tal y como se recogen en el capítulo 3 52
Tabla 12: Resumen de las actividades con sus fechas de inicio y duración 71
Tabla 13: Resumen de las actividades con el tiempo dedicado a cada una de ellas 72
Tabla 14: Presupuesto del personal 72
Tabla 15: Coste del material 72
Tabla 16: Presupuesto total 73
Tabla 17: Actividades del cliente 81

Acrónimos

API: Application Programming Interface

ASP: Active Server Pages

BSON: Binary JSON

DNS: Domain Name System

FTP: File Transfer Protocol

HTML: HyperText Markup Language

JSP: JavaServer Pages

JSON: JavaScrip Object Notation

MOOC: Massive open online course

OTRI: Oficina de Transferencia de los Resultados de la Investigación

RIM: Research In Motion

SHA-1: Secure Hashing Algorithm

SGDB: Sistema de gestión de bases de datos

W3C: World Wide Web Consortium

XML: eXtensible Markup Language

Capítulo 1: Introducción

En este capítulo se introducirá al lector en la temática relacionada con el proyecto. Se marcarán los objetivos y se explicará la estructura que seguirá toda la memoria con una breve descripción de cada capítulo.

1.1 Contexto

Actualmente miles de estudiantes tienen la oportunidad de acceder a recursos y contenidos educativos de alta calidad de forma gratuita. Esto es posible gracias a los MOOCs (*Massive Open Online Courses* - Cursos masivos abiertos online). Este tipo de cursos han supuesto una verdadera revolución en el sector de la educación.

Los MOOCs nacieron en la universidad de Stanford (otoño de 2011), donde uno de sus profesores decidió ofrecer de forma gratuita y a través de internet un curso para todas las personas que estuvieran interesadas. Con este curso el profesor consiguió 120.000 alumnos, todo un éxito que sirvió para iniciar una tendencia en el sector de la educación. Desde este momento los MOOCs no han parado de crecer, teniendo una evolución positiva durante los siguientes tres años [1].

Algunas de las plataformas de distribución de MOOCs más importantes son *Coursera, EdX, Udacity, FutureLearn o MiríadaX.* Estas plataformas no dejan de crecer y no se espera un estancamiento inmediato sino que se cree que el crecimiento será positivo ofreciendo cursos de temática cada vez más variada y en una cantidad de lenguas mayor. Un ejemplo de este éxito es *Coursera,* que cuenta ya con más de 8 millones de usuarios y más de 700 cursos de diferentes universidades [2].

Como se puede observar en la *ilustración 1* España es el país líder en materia de MOOC a nivel europeo. España abarca el 35% de los MOOCs ofrecidos en toda Europa (Diciembre 2013). Las universidades españolas son las principales causantes de esta gran oferta disponible, ya que más de un tercio de las universidades nacionales ofrecen al menos un MOOC. La plataforma de distribución de MOOCs más utilizada en España es *Miríada X,* donde más del 75% de las universidades españolas que ofrecen MOOCs han tenido presencia [3].

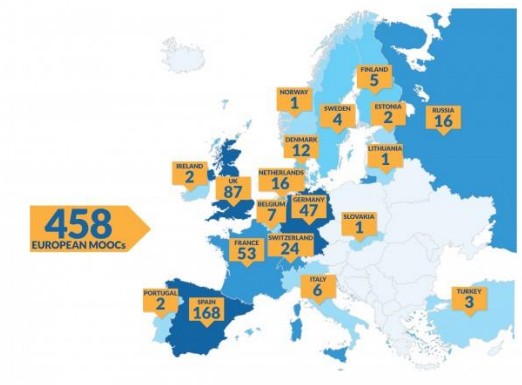


Ilustración 1: Distribución de los MOOCs en Europa (figura tomada de [4])

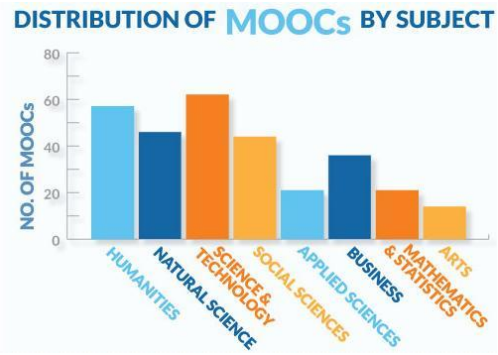


Ilustración 2: Distribución de los MOOCs por temática en Europa (Diciembre 2013) (Figura tomada de [4])

Uno de los principales factores que pueden encontrarse en el reciente éxito de los MOOCs en España es la situación socio-económica actual, donde existe más del 24% de tasa de desempleo (Julio 2014) [5]. Esta situación obliga a los ciudadanos a buscar nuevas oportunidades que les permitan acceder al mercado laboral. Muchas personas ven en los MOOCs una nueva oportunidad para aprender o actualizar sus conocimientos con el fin de incorporarse al mercado laboral [6].

En la *Ilustración 2* se muestra la distribución de los MOOCs por áreas temáticas en Europa. El área con mayor número de cursos ofrecidos es el de ciencia y tecnología. El segundo área con mayor cantidad de cursos ofrecidos es el de humanidades. Entre los que cuentan con menos oferta destaca el área de las artes.

1.2 Motivación

Precisamente que los MOOCs sean online y abiertos facilita que sean accesibles para todo tipo de personas, independientemente de su ubicación o situación económica. Esta gran variedad de posibles estudiantes hace que las personas que acceden a estos cursos sean de diferentes perfiles y les guíen diferentes motivaciones, dependiendo de las diversas circunstancias que cada una tenga en su vida.

Según lo expuesto en varios estudios [6], la mayoría de las personas que se matriculan en alguno de estos cursos tienen una edad de entre 25 y 40 años, poseen una carrera, un máster o un doctorado y tienen como objetivo actualizar o añadir diferentes áreas a sus conocimientos. Pero actualmente los MOOCs cuentan con un nuevo perfil de estudiante, gente sin empleo que busca adquirir algún tipo de habilidad que les permita acceder al mercado laboral. A diferencia de los estudiantes mayoritarios en los MOOCs, los alumnos con este perfil suelen tener mayores dificultades a la hora de ponerse a estudiar con éxito alguno de estos cursos, ya que puede que hayan perdido los hábitos de estudios o directamente nunca haberlos adquirido.

Al realizar un curso online surge un inconveniente que afecta de forma diferente a los diversos perfiles de estudiantes. Este inconveniente adicional consiste en la falta de contacto con el personal docente motivado por la gran cantidad de alumnos matriculados en cada curso. Para los estudiantes que no poseen hábitos de estudio, puede ser una tarea complicada el completar todas las tareas requeridas, de forma autónoma, y sin tener contacto con el personal docente del curso. De ser un curso presencial (y con un número menor de alumnos),

el personal docente hubiera podido proporcionar un guiado y una tutorización mayor en la realización de las correspondientes tareas. Como resultado puede surgir un sentimiento de desesperación que lleve al estudiante a abandonar el curso.

Tal y como se expone en el artículo *"Scaffolding self-learning in MOOCs"*, existen una serie de estudios que han demostrado que el trabajo en equipo y una buena organización del tiempo de estudio son esenciales para afrontar con éxito un curso online. Esto es fácil de conseguir en un curso presencial, pero puede convertirse en un problema de difícil solución cuando se trata de un curso online. Estas causas también contribuyen al abandono de los MOOCs, pero junto con estas causas existen otras como la falta de motivación para llevarlo a cabo y no estar realmente interesado en lo que el curso ofrece. Por todo ello se estipula que solo un 10% de personas finalizan los MOOCs que hubiesen iniciado [6].

Como solución a estos problemas se plantea *MyLearningMentor*, una aplicación que da soporte a la realización de MOOCs mediante un conjunto de funcionalidades pensadas para afrontarlos con garantías. *MyLearningMentor* controlará desde la cantidad de cursos a los cuales el usuario debería como máximo apuntarse hasta el tiempo que tarda en realizar cada tarea. Con esta información *MyLearningMentor* pretende ayudar a los estudiantes a afrontar un curso online, abierto y masivo.

1.3 Objetivos

El trabajo consiste en una primera implementación de *MyLearningMentor*, cubriendo buena parte de su funcionalidad. *MyLearningMentor* plantea una aplicación que ayude a los estudiantes de los MOOCs a finalizar con éxito los cursos. Los objetivos pueden ser resumidos en la siguiente lista:

1. Estudiar el contexto y la motivación para la realización de la aplicación.
2. Estudiar la solución planteada en el artículo donde se propone la aplicación [6].
3. Estudiar las diferentes tecnologías que existen actualmente con las que se podría realizar la aplicación y seleccionar las tecnologías más adecuadas.
4. Realizar un refinamiento del diseño de la aplicación.
5. Implementar *MyLearningMentor*.
6. Validar el prototipo implementado a partir de los requisitos funcionales planteados previamente.
7. Plantear las líneas futuras relacionadas con la continuidad en el desarrollo de la aplicación.

1.4 Estructura de la memoria

La memoria quedará dividida en cinco capítulos tratando cada uno de ellos un aspecto importante de la aplicación. En cada capítulo se tratarán temas claramente diferenciados aunque en algún momento puedan hacerse referencias cruzadas entre ellos con el fin de aclarar alguna situación. Cada capítulo tratará uno o varios de los objetivos anteriormente planteados, incluso algún en capítulo se agruparán dos objetivos.

En el primer capítulo se ha situado al lector en el contexto socio-económico actual. Se han introducido los MOOCs y las principales plataformas de distribución de MOOCs. Se ha realizado un análisis del impacto que éstos han tenido en la sociedad y se han estudiado los principales problemas que surgen de su utilización. Por último, se ha propuesto *MyLearningMentor,* una aplicación que intenta solucionar los problemas que surgen, principalmente, por tratarse de cursos online con un gran número de alumnos con perfiles heterogéneos (Objetivo 1).

El capítulo 2 estará dividido en dos partes. La primera de ellas se encargará de plantear una primera aproximación de la aplicación a partir del artículo en la que se propuso (Objetivo 2). En la segunda parte del capítulo se hará un estudio del estado del arte, planteando las principales tecnologías disponibles para llevar a cabo una aplicación como la planteada en la primera parte del capítulo. A partir de lo expuesto se decidirá con qué tecnologías se llevará posteriormente a cabo la implementación de *MyLearningMentor* (Objetivo 3).

El capítulo 3 también estará dividido en dos partes. La primera de ellas consistirá en un refinamiento de la aplicación, estudiando las diferentes variaciones que se realizarán con respecto a la arquitectura inicial anteriormente planteada en el capítulo 2. En esta sección se plantearán también los requisitos y las funcionalidades que la aplicación tendrá que cumplir al concluir el proyecto (Objetivo 4). En la segunda parte del capítulo se llevará a cabo la implementación del proyecto, y se mostrarán las diferentes partes de las que consta la aplicación tanto del lado del cliente como del lado del servidor. Además, para indicar con mayor claridad la implementación, se mostrarán tanto diagramas de clases como diferentes diagramas de secuencia (Objetivo 5).

En el capítulo 4 se llevará a cabo la validación de la aplicación, realizando diferentes pruebas para comprobar que todos los requisitos anteriormente planteados se cumplen (Objetivo 6). Además en este capítulo se explicará cómo se ha protegido la información más delicada del usuario.

Por último en el capítulo 5 se presentarán las conclusiones extraídas de la realización del proyecto y las líneas futuras de este trabajo, con el fin de mejorar el rendimiento de las funcionalidades existentes o crear otras nuevas para que la aplicación sea cada vez más completa (Objetivo 7).

Al final de la memoria se incluirán cuatro anexos. El primer anexo estará dedicado a la planificación de las diferentes tareas necesarias para concluir el trabajo. Las tareas se expondrán gráficamente mediante un diagrama de Gantt. El segundo anexo incluirá el presupuesto desglosado del proyecto. Para la realización del presupuesto se tendrán en cuenta tanto los gastos materiales como los gastos de personal. El tercer anexo presentará el marco legal donde se sitúa *MyLearningMentor*. Para analizar el marco legal se tendrán en cuenta las legislaciones necesarias a nivel español y europeo. En el cuarto anexo se presentará un resumen de todas las actividades de la parte del cliente, como se explicará con más detalle en el capítulo 4.

CAPITULO 2: MyLearningMentor

Este capítulo se dividirá en dos partes. En la primera parte se explicará la arquitectura inicial de *MyLearningMentor*. En la segunda parte se realizará un estudio del estado del arte. Se expondrán las diferentes tecnologías que podrían ser usadas para implementar la arquitectura y se elegirán las tecnologías más adecuadas para la implementación de *MyLearningMentor*.

2.1 MyLearningMentor

En esta sección se hará un primer acercamiento a *MyLearningMentor* según lo expuesto en el artículo "*Scaffolding self-learning in MOOCs*" [6]. Esta aproximación servirá de guía en capítulos posteriores donde se analizará más detalladamente la aplicación y se llevará a cabo su implementación.

2.1.1 Introducción

MyLearningMentor es una propuesta de una arquitectura con un conjunto de requisitos que posteriormente serán implementados. Con esta aplicación se pretende ofrecer a los diferentes perfiles de estudiantes un apoyo en la realización de un MOOC. Como ya se ha comentado en el capítulo anterior existe una heterogeneidad de perfiles en los usuarios que acceden a un MOOC. Esta diversidad es debida a que no todos los usuarios han adquirido previamente los mismos conocimientos ni tienen los mismos hábitos de estudio. Como solución a los problemas mencionados, *MyLearningMentor* plantea un sistema de organización que ayuda a los estudiantes a finalizar con éxito un curso online, ya que como se ha indicado anteriormente solo un 10% de los estudiantes lo consiguen [6].

La arquitectura planteada en el artículo *"Scaffolding self-learning in MOOCs"* [6] se ha realizado a partir de unos requisitos que debería cumplir la aplicación final. Estos requisitos están resumidos en la *tabla 1*.

El primer requisito (Requisito 1) es que se debe tratar de una aplicación móvil. Esta elección se justifica en la medida en que la mayoría de las personas que típicamente acceden a un MOOC tienen una edad de entre 25 y 40 años y utilizan un teléfono móvil de forma continuada en su vida diaria [6].

Identificador	**Requisitos**
Requisito 1	Distribuirse como una aplicación móvil.
Requisito 2	Adaptarse según los diferentes perfiles de estudiante.
Requisito 3	Ofrecer un planificador diario.
Requisito 4	Confiar en la información facilitada por los usuarios sobre los MOOCs.
Requisito 5	Proporcionar consejos y sugerencias para aprovechar al máximo los MOOCs.
Requisito 6	Servir como punto de encuentro entre estudiantes y tutores voluntarios.

Tabla 1: Requisitos de la aplicación

El segundo requisito (requisito 2) consiste en que la aplicación debe adaptarse al perfil de cada usuario. Como se ha comentado anteriormente no existe un único perfil de usuario sino que existe una diversidad de perfiles que dependerá cada uno de ellos de las circunstancias personales de cada usuario. Para poder dar un servicio de calidad al usuario es importante que la aplicación tenga este requisito en cuenta ya que de lo contrario la aplicación carecería de sentido al poder ser utilizada solo por un grupo concreto de personas. El tercer requisito (Requisito 3) es que la aplicación debe ofrecer al usuario un planificador según su perfil. En este planificador se listarán ordenadas las tareas que debe realizar según los cursos a los que se haya unido previamente. El cuarto requisito (Requisito 4) es que la aplicación debe confiar en la información que los usuarios facilitan sobre los MOOCs. Este requisito es importante ya que aunque se consiguiera implementar una aplicación que extrajera la información de los cursos automáticamente de sus respectivos sitios web, existiría información que sería demasiado difícil de extraer directamente, como todas las tareas con sus correspondientes fechas de entrega. Por esta razón son los usuarios los que deben completar esta información. Y por tanto la aplicación debe fiarse de la información introducida por los usuarios para su correcto funcionamiento, como por ejemplo para calcular el orden de las tareas facilitado a los usuarios en el planificador. El quinto requisito (Requisito 5) consiste en que la aplicación debe proporcionar consejos y sugerencias a los usuarios. Estos consejos y estas sugerencias variarán según el perfil de cada estudiante y tendrán como fin ayudar a mejorar las conductas y hábitos de estudio de los usuarios. Por último el sexto requisito (Requisito 6) es que la aplicación sirva como punto de encuentro para estudiantes y personas voluntarias que ayuden a estos estudiantes con los problemas que les puedan surgir durante la realización de los cursos.

2.1.2 Arquitectura

A continuación se va a explicar la arquitectura propuesta de *MyLearningMentor* para la implementación de la aplicación, según lo expuesto en el artículo "*Scaffolding self-learning in MOOCs*" [6]. La aplicación será diseñada para cumplir con todos los requisitos expuestos anteriormente en la *tabla 1*. Como se observa en la *Ilustración 3*, en *MyLearningMentor* se plantea una arquitectura cliente-servidor para dispositivos móviles. La elección de este tipo de arquitectura ha sido motivada por dos razones. La primera de estas razones es que los usuarios accederán a la aplicación a través de un dispositivo móvil (Requisito 1 de la *tabla 1*). La segunda de estas razones es que los usuarios tendrán que acceder a información que otros usuarios vayan añadiendo sobre los cursos (Requisito 4 de la *tabla 1*). *MyLearningMentor* es una arquitectura que está propuesta inicialmente para su implementación como aplicación móvil en un dispositivo Android. Sin embargo, antes de llevar a cabo la implementación de la arquitectura se estudiarán los diferentes sistemas operativos disponibles actualmente en el mercado para valorar todas las posibilidades y elegir las más adecuada en este momento. De igual forma se estudiarán todas las posibles variantes para la implementación del lado del servidor. Cuando se hayan elegido todas las tecnologías a utilizar se podrá iniciar la implementación de la aplicación.

En la *Ilustración 3* se observa que existen cuatro servicios principales que ofrece *MyLearningMentor*. Estos servicios son: *Cuenta y gestión de perfiles (Account & Profile Management)*, *Directorio de MOOCs (Mooc Directory)*, *Actividades sugeridas de los MOOCs (MOOC Activity Suggestions)* y *Recopilación de información (Feedback Gathering)*. El servicio

Cuenta y gestión de perfiles facilitará a un estudiante la posibilidad de registrarse como usuario de la aplicación y controlará el acceso a la misma. Además será el que permita añadir y modificar los perfiles a los usuarios. El servicio *Directorio de MOOCs* será el encargado del tratamiento de la información de los diferentes MOOCs. Permitirá el almacenamiento de información relativa a los cursos y además controlará la cantidad de cursos a los que el usuario puede apuntarse según su disponibilidad. *Actividades sugeridas de los MOOCS* será el servicio encargado de ofrecer al usuario un planificador personal según su perfil académico. El último servicio es *Recopilación de información* que es el encargado de obtener la información relativa a las tareas de los cursos que los usuarios van introduciendo [6].

En la *Ilustración 3* también se muestran las diferentes bases de datos de información. En la arquitectura inicial se muestran tres bases de datos diferentes: *perfiles de usuarios (User Profile), información de los MOOCs (MOOCs Data) y retroalimentación (Feedback).* La base de datos *perfiles de usuarios* almacenará la información relacionada con el perfil de los usuarios almacenando datos como nombre, email y contraseña. La base de datos *información de los MOOCs* contendrá la información de los cursos como URL, nombre y duración. Por último la tercera base de datos, *Retroalimentación,* almacenará información sobre los progresos de los estudiantes en los cursos [6].

En la *Ilustración 3* se muestran además dos procesos que están ejecutándose periódicamente en el servidor. El primer proceso es *recopilación de datos (Gathering MOOC data). MyLearningMentor* recogerá de las plataformas de distribución de MOOCs información de los cursos como: la duración del curso, el tiempo de dedicación recomendado por semana y el tipo de las actividades. El segundo proceso es *definición de las actividades recomendadas (Difining activity recommendations). MyLearningMentor* definirá las tareas recomendadas para cada usuario teniendo en cuenta su perfil y los datos de los MOOCs [6].

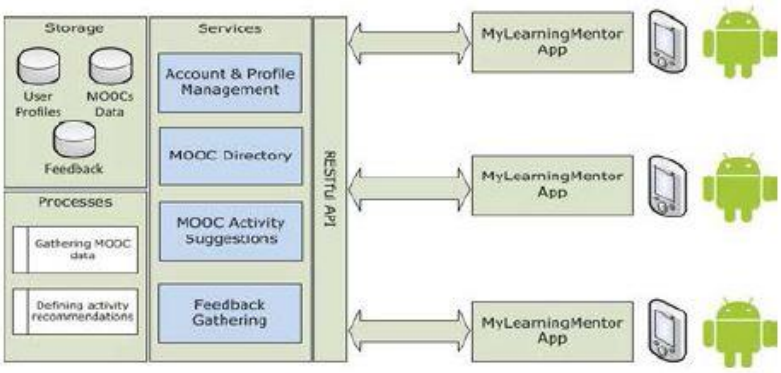


Ilustración 3: Arquitectura de MyLearningMentor (figura tomada de [6])

2.1.3 Interfaz

En esta sección también se va a plantear una primera aproximación de las diferentes interfaces que se le mostrarán al usuario tal y como se recoge en el artículo "*Scaffolding self-learning in MOOCs*" [6]. En la *Ilustración 4* se muestran tres capturas de pantallas con posibles interfaces de la aplicación en tres momentos diferentes.

Lo primero con lo que se encontrará cualquier usuario al acceder a la aplicación será con una pantalla de bienvenida donde podrá introducir su email y contraseña. En el caso de que el usuario todavía no esté registrado en la aplicación se le permitirá acceder a una pantalla de registro desde esta pantalla inicial. Una vez que el usuario haya accedido a la aplicación podrá pasar a modificar sus perfiles, tanto su perfil personal (*Ilustración 4a*) como su perfil de estudiante (*Ilustración 4b*). Una vez que ha cumplimentado los ajustes de perfil será redirigido a una pantalla donde seleccionará los MOOCs a los que quiere unirse. La aplicación le ofrecerá la posibilidad de apuntarse a una serie de cursos que ya tiene almacenados en el sistema (*Ilustración 4c*). En el caso de que el usuario no encuentre el curso al que está matriculado, la aplicación le ofrecerá la posibilidad de añadir un nuevo curso. Cuando el usuario haya indicado los cursos en los que está matriculado podrá acceder al planificador el cual le ofrecerá una lista con las tareas pendientes. Esta lista de tareas se realizará según la disponibilidad semanal del usuario. En el planificador se permitirá al usuario marcar como completada una tarea, consiguiendo así que las tareas realizadas no vuelvan a mostrarse la próxima vez que el usuario acceda al planificador. Además de todo esto, el usuario tendrá la oportunidad de acceder a una lista de consejos cuando se encuentre con dificultades en los cursos en los que se haya matriculado.

Ilustración 4: Capturas de pantallas de MyLearningMentor. De izquierda a derecha: (a) Ajustes personales, (b) Perfil de estudiante y (c) Selección de cursos (Figura tomada de [6])

2.2 Estudio y elección de las tecnologías

De acuerdo con las necesidades de *MyLearningMentor* en esta sección se hará una exposición de las diferentes tecnologías que pueden ser utilizadas en el desarrollo de una aplicación para un sistema móvil inteligente (*smartphone*). Se hará un análisis de los diferentes sistemas operativos móviles que se encuentran actualmente en el mercado y de las estructuras que pueden seguir las aplicaciones para dichos sistemas operativos. Una vez elegida la estructura que parece más adecuada para este caso, se expondrán los diferentes lenguajes utilizados en cada una de sus partes y los posibles sistemas de almacenamiento de información.

2.2.1 Sistemas operativos móviles inteligentes.

En la actualidad existen varias opciones en el mercado de sistemas operativos para móviles. Debido a la gran cuota de mercado que los *smartphones* ocupan actualmente han adquirido

gran importancia tanto en el ámbito comercial como en el ámbito de la investigación, llegando hasta el punto de convertirse en sistemas muy parecidos a un ordenador personal [7].

En la *Ilustración 5* se observa cómo las ventas de los *smartphones* han aumentado de forma casi exponencial desde que salieran al mercado. Aunque los *smartphones* tardaron alrededor de 3 años en convertirse en un producto con una gran demanda, desde entonces sus ventas se han disparado. Como se refleja en la *Ilustración 5*, las ventas de los terminales iPhone junto con los *smartphones* con sistema operativo Android superan de forma muy significativa a las ventas de los PCs.

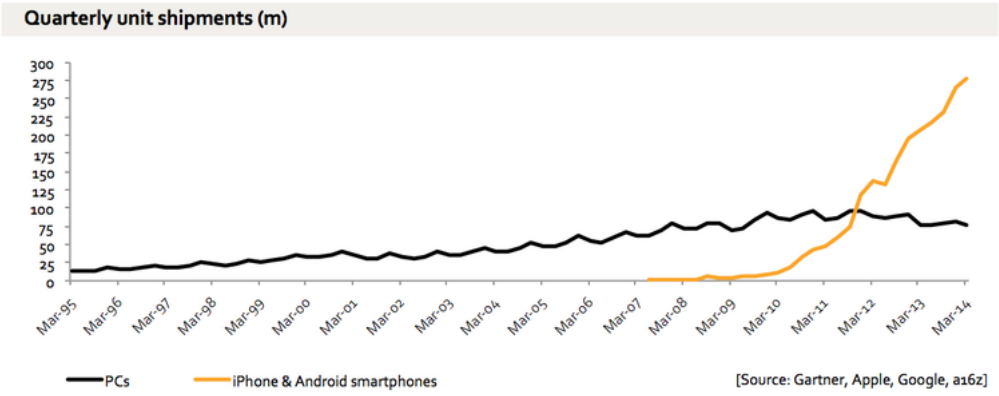


Ilustración 5: Evolución de PCs, iPhone & Android smartphones y tablets (Figura tomada de [8])

2.2.2 Ejemplo de sistemas operativos móviles

Existe una gran variedad de sistemas operativos móviles, entre los que se encuentran: Android, iOS, Symbian, Windows Phone, BlackBerry, Java ME, Kindle. En la *Ilustración 6* se muestran las cuotas de los diferentes sistemas operativos instalados en los nuevos *smartphones* durante el primer trimestre del año 2014. Se muestra la información detallada para diferentes lugares geográficos: España, Europa, EEUU, China, Australia y Japón. Como se puede ver en la *Ilustración 6,* la cuota del sistema operativo Android en España supera de forma notable a la de la media europea. El principal perjudicado es el sistema operativo iOS que en España reduce su cuota en casi 12 puntos porcentuales en comparación a la media europea. En la *Ilustración 6* también se puede apreciar la diversidad de preferencias existentes entre las diferentes zonas geográficas. Mientras que en China Android ocupa el 80.0% de la cuota de los nuevos dispositivos, en Japón esta cifra queda reducida al 41.5%. Además, en la *Ilustración 6* destaca la poca cuota que ha tenido el sistema operativo de BlackBerry en el primer trimestre de 2014.

	ESPAÑA	EUROPA*	EEUU	CHINA	AUSTRALIA	JAPÓN
Android	88,6	70,7	57,6	80,0	57,3	41,5
BlackBerry	0,0	1,1	0,7	0,1	1,0	0,0
iOS	7,6	19,2	35,9	17,9	33,1	57,6
Windows	3,0	8,1	5,3	1,0	6,9	0,9
Other	0,8	0,9	0,4	1,0	1,7	0,0

Fuente: Worldpanel Comtech
*Alemania, Reino Unido, Francia, Italia, España

Ilustración 6: Cuotas de los sistemas operativos sobre nuevos smartphones Enero-Marzo 2014 (figura tomada de[9])

Windows Phone

Windows Phone es el sistema operativo móvil desarrollado por Microsoft. Windows Phone surgió como sustituto de Windows Mobile. En primer lugar se diseñó Windows Phone 7, aunque actualmente ya está disponible Windows Phone 8 [10].

Windows Phone está pensado para un consumo generalista en vez de para un consumo empresarial, como era el caso de Windows Mobile. Ambos sistemas tienen muchas características comunes, sin embargo, Microsoft hizo que las aplicaciones para ambos sistemas operativos no fueran compatibles entre ellos. Como consecuencia se obligó a una entera renovación del sistema operativo forzando a los desarrolladores a crear nuevas aplicaciones [10].

La primera versión de Windows Phone 7 se ofreció al público en septiembre del 2010. Hasta llegar al actual Windows Phone 8, se ha llevado a cabo una evolución pasando por diferentes versiones [10]:

- Windows Phone 7 (Photon)
- Windows Phone 7.5 (Mango)
- Windows Phone 7.5 (Tango): reduce los requisitos de hardware necesarios para que un dispositivo pueda soportar Windows 7.
- Windows Phone 7.8: pensada para teléfonos que no son actualizables a Windows Phone 8.
- Windows Phone 8

BlackBerry OS

El sistema operativo BlackBerry OS fue creado por la empresa RIM (*Research in Motion*). BlackBerry OS tiene sus inicios en 2002 cuando RIM anunció el lanzamiento del dispositivo BlackBerry 5810 el cual llevaría consigo la primera versión del sistema operativo BlackBerry OS.

El sistema operativo no tiene demasiado éxito ya que los terminales con los que se ofrecía no eran realmente comerciales. En el año 2006 BlackBerry pone a la venta dos terminales completamente comerciales. En esta versión del sistema operativo se introduce el *BlackBerry App World* con la misión de competir con las otras tiendas de aplicaciones del mercado. Fue durante este periodo cuando este sistema operativo alcanzó su mayor éxito. BlackBerry no ha

sabido actualizar su sistema operativo y esto ha llevado a que actualmente la cuota de mercado de BlackBerry OS sea insignificante [11].

La evolución del sistema operativo se ha llevado a cabo por las diferentes versiones [12][13]:

- RIM OS 1.0
- OS 3.x
- OS 4.x
- OS 5
- OS 6
- OS 7
- BlackBerry 10

iOS

Apple revela la existencia de su primer sistema operativo móvil por primera vez en Enero de 2007. El sistema operativo creado por Apple se ofrecía junto con el dispositivo iPhone, el teléfono móvil que lo soportaría. En esta primera versión todavía no se le dio un nombre al nuevo sistema operativo móvil aunque más tarde se le bautizaría como iOS. El sistema operativo iOS solo puede ser utilizado en los dispositivos móviles de la compañía Apple [14].

Apple está presente en todas las fases de desarrollo de su dispositivo, desde la implementación del propio terminal, hasta el software que permite instalar a los usuarios en sus dispositivos. Con este control Apple consigue el dominio total del sistema operativo iOS. El lenguaje de programación necesario para programar en iOS es Objetive C, que es un lenguaje mucho menos conocido que otros como Java o C. Para desarrollar una aplicación en iOS es necesario disponer de un ordenador personal de Apple. Si se quiere subir la aplicación al mercado de aplicaciones se debe de pagar una cuota anual. Además, antes de que una aplicación sea subida tiene que ser aprobada por Apple [15].

La evolución que ha seguido el sistema operativo creado por Apple se muestra a continuación [16][17]:

- iPhone OS 1
- iPhone OS 2
- iPhone OS 3
- iOS 4
- iOS 5
- iOS 6
- iOS 7

Android

El sistema fue desarrollado por la empresa Android Inc., la cual fue creada en 2003. Esta empresa estuvo respaldada económicamente por Google y fue posteriormente adquirida en 2005 por esta misma empresa. Se trata de un sistema operativo basado en Linux, cuya principal idea es la creación de software para dispositivos móviles de pantalla táctil. Android es

de código abierto (exceptuando mínimas cosas propiedad de Google) bajo la licencia Apache [18].

En la fabricación de un dispositivo con sistema operativo Android intervienen tres actores: Google, los desarrolladores de hardware y los creadores de software. Con esto se divide la implementación de un nuevo dispositivo y no se entrega el control exclusivamente a un único actor. El lenguaje de programación utilizado en Android es Java. El desarrollo de una aplicación Android solo supone un coste fijo a la hora de introducir la aplicación en el mercado de aplicaciones. Esto se debe a que el entorno de desarrollo es totalmente gratuito y se puede utilizar en diferentes sistemas operativos como Windows, MacOS (sistema operativo de los ordenadores Apple) y Linux. Una vez que se ha realizado la aplicación se puede subir al mercado de aplicaciones sin que ésta sea revisada previamente por ningún miembro de Android [15].

Desde que llegara la primera versión de Android en 2007 el sistema operativo ha seguido una evolución por medio de las diferentes versiones. Esta evolución se muestra a continuación [19]:

- Android 1.0, Apple Pie
    - Android 1.1, Banana Bread
    - Android 1.5, Cupcake
    - Android 1.6, Donut
- Android 2.0, Eclair
    - Android 2.1
    - Android 2.1(Multitouch)
    - Android 2.2 FroyoAndroid 2.3 Gingerbread
- Android 3.0, Honeycomb
- Android 4.0, Ice Cream Sandwich
    - Android 4.1, Jelly Bean
    - Android 4.2
    - Android 4.3
    - Android 4.4, KitKat

Elección del sistema operativo

En el momento en el que se desarrolla la aplicación las dos versiones de los sistemas operativos con una mayor cuota de mercado son: iOS 7 y Android 4.4 KitKat. En la *tabla 2* se muestra una comparativa entre los dos sistemas operativos iOS y Android realizada a partir lo expuesto en secciones anteriores. La elección final del sistema operativo en la que se realizará la aplicación siguiendo la arquitectura de *MyLearningMentor* ha sido Android. La decisión de utilizar Android, en vez de iOS, ha venido motivada por diversas razones expuestas a continuación [15].

- La mayor cuota de mercado que tiene Android sobre iOS en la mayoría de los países [9].
- La facilidad que aporta Android en la implementación de la aplicación. Android ofrece sus herramientas de desarrollo disponibles para varios sistemas operativos de

ordenadores. Mientras que para iOS solo es posible desarrollar aplicaciones desde ordenadores de marca Apple.

- El conocimiento sobre el lenguaje de desarrollo que se utiliza para la implementación de una aplicación Android. En Android el lenguaje utilizado es Java que es un lenguaje ya conocido. En iOS es *Objetive C,* mucho menos conocido.
- La comodidad para subir al mercado de aplicaciones la aplicación una vez que esté finalizada. En iOS previamente tiene que ser aprobada por Apple mientras que Android esta aprobación no es necesaria.
- El precio total del proyecto es menor en Android al tener que pagar solo una cuota fija al subir la aplicación al mercado de aplicaciones. En iOS habría que pagar una cuota anual.

	Android	**iOS**
Cuota de mercado	Mayor en Europa, EEUU, China y Australia	Mayor en Japón
Herramientas de desarrollo	Windows, MacOS y Linux	MacOS
Lenguaje	Java	Objetive C
Subir aplicación	Sin revisión previa	Con revisión previa
Precio	Solo una vez	Anualmente

Tabla 2: Comparativa Android e iOS

2.2.3 Arquitecturas de sistemas distribuidos.

En un sistema distribuido el procesamiento de información se lleva a cabo en varios sistemas, en vez de llevarse a cabo en el mismo sistema. Este tipo de arquitecturas tienen una serie de ventajas comentadas a continuación [20].

- **Posibilidad de compartir los recursos**: permitiendo la posibilidad de compartir recursos hardware y software que se asocian dentro de una red.
- **Apertura**: se trata de sistemas en su mayoría abiertos diseñados sobre protocolos estándar, permitiendo así la combinación de productos de diferentes fabricantes.
- **Escalabilidad**: se trata de sistemas a los que se les puede añadir nuevos sistemas para cubrir mayores demandas. El número máximo de recursos que se puede llegar a añadir puede estar limitado.
- **Tolerancia a fallos**: permitiendo la posibilidad de usar otros sistemas en el caso de que uno falle.

Las desventajas que se pueden encontrar en este tipo de sistemas son las siguientes.

- **Complejidad**: son más difíciles de gestionar que los sistemas centralizados.
- **Seguridad**: si el tráfico tiene que pasar por varias máquinas pueden producirse escuchas en el intercambio de información y por tanto la seguridad verse comprometida.
- **Manejabilidad**: las diferentes máquinas que forman esta estructura pueden ser de diferentes tipos y cada una correr en un sistema operativo diferente.
- **Impredecibilidad:** la respuesta del sistema puede cambiar dependiendo de las circunstancias.

Las arquitecturas que pueden tener los sistemas distribuidos se describen brevemente a continuación [20].

- **Multiprocesador:** el software está formado por varios procesos que pueden ejecutarse sobre procesadores diferentes.
- **Cliente-Servidor:** el sistema se encarga de proporcionar una serie de servicios a los diferentes clientes.
- **Objetos distribuidos:** no distingue entre servidores y clientes. No existe una diferencia entre un posible proveedor de servicio y un posible cliente.
- **Peer to peer:** son sistemas descentralizados en donde los cálculos se pueden llevar a cabo en cualquier nodo de la red. En principio este tipo de arquitectura no hace distinciones entre clientes y servidores.
- **Orientada a servicios:** quiere proveer de un servicio independientemente de cuál sea la aplicación que use dicho servicio.

En el caso de *MyLearningMentor* se ha utilizado el modelo cliente-servidor, ya que en este caso diferentes usuarios realizarán peticiones independientes a un servidor, el cual solo esperará peticiones de usuarios de *MyLearningMentor*.

Cliente-Servidor

La comunicación en una arquitectura cliente-servidor consiste en un conjunto de peticiones realizadas en orden cliente-servidor, de forma que el servidor es el encargado de atender las peticiones que realiza el cliente. *MyLearningMentor* seguirá una estructura semejante a la mostrada en la *Ilustración 7* que es la arquitectura habitualmente utilizada para aplicaciones que tienen que extraer datos de una base de datos [20].

Ilustración 7: Ejemplo de arquitectura cliente-servidor (cliente con sistema operativo Android) (Figura tomada de [21])

Servidor

El servidor es un ordenador o máquina informática que se encarga de suministrar todo tipo de información requerida por un cliente. Los servidores pueden tener diferentes funciones y dependiendo de éstas serán denominados de una forma u otra [22].

- **Proxy**: hace de intermediario entre dos ordenadores. Al usar este tipo de servidor se consigue que el destinatario no conozca quién le está realizando la petición. También puede bloquear el acceso a determinadas peticiones.
- **DNS (*Domain Name Server*):** es un tipo de servidor que asocia un nombre de dominio a una cierta información.
- **WEB:** el servidor hace que los clientes puedan acceder a las diferentes páginas web de la red.
- **FTP (*File Transfer Protocol*):** permite la transferencia de archivos entre cliente y servidor, permitiendo por ejemplo que el cliente pueda descargarse un archivo del servidor o el cliente puede mandar un archivo al servidor.

2.2.4 Sistemas de gestión de bases de datos

Los SGDB (sistemas de gestión de bases de datos) se suelen basar en el modelo relacional y son los que permiten almacenar datos para posteriormente acceder a estos de forma rápida y estructurada. La elección del SGDB se va a dividir en dos primeras opciones para que una vez elegida una de éstas se elija el SGBD a utilizar en la aplicación [23][24][25].

- **SGBD libres**: son aquellos SGDB que están bajo una licencia libre y por tanto su uso, modificación y distribución son permitidos para todos los usuarios.
- **SGBD comerciales**: son aquellos que están bajo una licencia comercial y que en muchas ocasiones es necesario pagar para su uso.

En este caso se ha decidido usar software que esté bajo una licencia libre, ya que así no se tendrán problemas de licencia durante el desarrollo de la aplicación.

Dentro de la clase de software libre se encuentran diferentes opciones. A continuación se indicarán las más comunes. Dentro de estas dos opciones la aplicación se desarrollará en alguno de los sistemas de gestión de base de datos que trabajen en entornos cliente-servidor.

- **PostgreSQL**: es un sistema de gestión de bases de datos objeto-relacional. Utiliza un modelo cliente-servidor. Dentro de las principales ventajas que se encuentran en PostgreSQL destacan algunas como: ser gratis y estable, permitir realizar consultas complejas y tener control de concurrencia. Entre sus principales desventajas se encuentran la necesidad de tener administradores capacitados y tener una velocidad inferior a otros sistemas [26].
- **MySQL**: es el sistema de gestión de bases de datos más popular. *MySQL* trabaja en entornos clientes-servidor. Entre las principales ventajas que se pueden encontrar en *MySQL* destacan: el poder llevar a cabo su uso sin necesidad de mucha memoria RAM, la posibilidad de ejecutar *MySQL* en una máquina con pocos recursos, ser fácil de instalar y configurar, tener una buena integración con PHP, poseer buenas herramientas de administración, tener pocas probabilidades de corromper datos, permitir agrupación de transacciones y ofrecer una conectividad segura. Entre sus principales desventajas destacan: no existir la posibilidad de sincronizar datos con otras bases de datos réplicas y no existir la posibilidad de autentificación local [26][27].

Para el desarrollo de esta aplicación se ha decidido usar ***MySQL***, teniendo en cuenta varios factores como: la buena integración con los principales lenguajes del lado del servidor (por ejemplo PHP), tratarse de un sistema muy popular que facilita la adquisición de información y documentación y por la existencia de buenas herramientas de administración las cuales facilitan el uso de las bases de datos.

2.2.5 Sistemas de gestión documental

En este sistema de base de datos cada registro queda almacenado como un documento con un identificador único al cual se puede acceder desde distintos campos. Existen varias opciones dentro de las bases de datos documentales [28][29][30][31]:

- **Cassandra**: puede ejecutarse como un nodo simple y como un servidor distribuido. Dentro de sus características más importantes cabe destacar varias como escalabilidad horizontal, rapidez de respuesta, almacenamiento distribuido, detección automática de fallos, tolerancia a fallos, la no existencia de un fallo único. Como principales inconvenientes de esta tecnología se encuentran la imposibilidad de utilizar *joins* (combinar datos que no están almacenados en la misma tabla) y la incapacidad para ordenar resultados en tiempo de consulta.

- **CouchDB**: se trata de una base de datos orientada a documentos que es accesible mediante una API (*Application Programming Interface*) que hace uso de JSON. Las principales características que se buscan con este sistema son la capacidad de ser muy escalable y de contar con una alta disponibilidad y robustez. Su principal inconveniente es la imposibilidad de usar *joins* y campos *auto-increment* (campos únicos creados automáticamente al insertar un nuevo registro en la base de datos).

- **MongoDB**: es un sistema de base de datos orientado a documentos que guarda estructuras de datos documentales tipo BSON (*Binary JSON*). El almacenamiento de datos se lleva a cabo mediante un esquema dinámico que hace que la integración de los datos en las aplicaciones sea rápida y fácil. La estructura del documento es simple y compuesta por pares clave-valor.

La elección en este caso será ***MongoDB*** para aprovechar la facilidad que nos ofrece al guardar de forma rápida los pares clave-valor. Además también posee integración con lenguajes del lado del servidor (por ejemplo PHP), lo que facilita el acceso a las bases de datos desde el servidor [32].

2.2.6 Lenguajes de parte del servidor

Se trata de lenguajes que se ejecutan del lado del servidor y éstos pueden tener acceso a las bases de datos. Los lenguajes del lado del servidor más utilizados son: ASP, JSP, PERL y PHP [33][34].

- **ASP (*Active Server Pages*)**: Consiste en un conjunto de clases que son utilizadas para crear aplicaciones Web, y que se utilizan tanto del lado del cliente como del lado del servidor. Entre las principales ventajas que encontramos son que se trata de un

lenguaje completamente orientado a objetos, es seguro y rápido, y de fácil mantenimiento para grandes aplicaciones. Su principal desventaja es que tiene un consumo elevado de recursos.

- **JSP (*JavaServer Pages*)**: Se trata de un lenguaje para la creación de sitios web dinámicos, orientado a desarrollar páginas web en Java. Entre sus principales ventajas se encuentran: permitir una ejecución rápida de *servlets* (código Java utilizado por un servidor), tener el código bien estructurado y ser multiplataforma. Su principal inconveniente es la dificultad que lleva su aprendizaje.
- **PERL**: Se trata de un lenguaje de programación usado para la administración de tareas propias de sistemas UNIX. Se caracteriza porque no establece ninguna filosofía de programación concreta. Es un lenguaje de programación que es portable casi a cualquier plataforma. Ofrece varias ventajas entre las que se pueden encontrar: ser un sistema rápido y aportar facilidades para juntar varios programas.
- **PHP**: Se trata de un lenguaje de programación del lado del servidor, rápido, independiente de la plataforma y gratuito. El cliente recibe una página con código HTML (*HiperText Markup Language*). Es compatible con todos los navegadores. Las ventajas de este lenguaje son: es fácil de aprender, soporta la orientación a objetos, es multiplataforma, tiene capacidad de conexión con *MySQL*, tiene una gran cantidad de funciones y no necesita manejo detallado de bajo nivel. Entre sus principales desventajas se encuentran: todo el trabajo lo ejecuta el servidor (lo que puede llegar a ser ineficiente) y hace difícil la modularización y la organización por capas de la aplicación.

Se utilizará como lenguaje en el lado del servidor **PHP**, por diversas razones. Se trata de un sistema rápido y que tiene la posibilidad de conectarse con las bases de datos ya elegidas (*MySQL* y *MongoDB*). Además se trata de un lenguaje de fácil aprendizaje y con mucha documentación en la web, tanto por parte de la página oficial, donde también vienen ejemplos, como de terceros. También se ha elegido este lenguaje para aprender a manejar un nuevo tipo de lenguaje de programación en el lado del servidor.

2.2.7 Comunicación entre cliente servidor.

Es necesaria la definición de un lenguaje de programación que nos facilite el intercambio de información entre cliente y servidor. Entre los más comunes tenemos.

- **XML (*eXtensible Markup Language*)**: Se trata de un lenguaje diseñado para transportar datos. Las etiquetas XML no están predefinidas sino que se puede definir etiquetas en función de las necesidades de cada programa. Se trata de una recomendación del W3C (*World Wide World Consortium*) [35].
- **YAML**: Se trata de un lenguaje de programación serializable de datos legibles por humanos [36].
- **JSON (*JavaScript Object Notation*)**: Se trata de un formato de intercambio de datos ligeros de fácil lectura y escritura. JSON está basado en dos estructuras: la primera una colección de pares nombre/valor; la segunda, una lista ordenada de valores [37].

Parece lógico, ya que el sistema de gestión de bases de datos documental elegido ha sido *MongoDB*, que el lenguaje de intercambios utilizado sea **JSON**. Además PHP ofrece buena integración para crear JSON lo que facilita el intercambio de información entre cliente y servidor. Con esta elección todo queda relacionado. El cliente mandará al servidor la información en JSON. El servidor obtendrá la información mandada por el cliente y la utilizará al comunicarse con las bases de datos, donde la información se almacena en formato clave-valor. Por último el servidor creará la respuesta en JSON, gracias a la buena integración entre PHP y JSON, y el cliente recibirá la respuesta en el mismo formato en el que envió la petición.

2.3 Conclusiones

La arquitectura *MyLearningMentor* será desarrollada en una aplicación para el sistema operativo Android. La elección final se ha realizado entre los dos sistemas operativos móviles con mayor cuota de mercado: Android e iOS. *MyLearningMentor* seguirá una arquitectura cliente-servidor ya que es la más adecuada al desarrollarse una aplicación para un sistema móvil. Los sistemas de almacenamientos de datos utilizados serán *MongoDB* y *MySQL*. El lenguaje utilizado en el lado del servidor será PHP y el lenguaje elegido para la comunicación entre cliente y servidor será JSON.

Capítulo 3: Refinamiento del diseño e implementación

En este capítulo se tratarán los temas relacionados con el refinamiento del diseño y la implementación de la aplicación *MyLearningMentor.* En la primera parte del capítulo se abordarán los requisitos definido en "*Scaffolding self-learning in MOOCs*" [6]. A partir de ellos, se hará un refinamiento del planteamiento inicial de la arquitectura de *MyLearningMentor.* Con este refinamiento se quiere conseguir una adaptación a las necesidades funcionales y a las decisiones de diseño según las tecnologías elegidas de acuerdo con lo establecido en el capítulo 2 de este documento. En la segunda parte del capítulo se presentará la implementación de la aplicación, tanto del cliente como del servidor. Se mostrarán los diferentes diagramas de clases y de secuencia para entender el funcionamiento interno de la aplicación.

3.1 Refinamiento del diseño

A partir de la arquitectura expuesta en el capítulo 2 se hará un refinamiento de la misma para obtener un diseño final de la arquitectura. El diseño final de la arquitectura *MyLearningMentor* obtenida en esta sección será la arquitectura finalmente desarrollada como aplicación móvil.

3.1.1 Requisitos

Partiendo de los requisitos de *MyLearningMentor,* planteados en el capítulo 2, se ha realizado un refinamiento de los mismos para poder abárcalos en este trabajo fin de grado. Estos nuevos requisitos serán los que se tendrán en cuenta a la hora de implementar la aplicación.

- Se tratará de una **aplicación móvil**, ya que se sabe que la gran mayoría de las personas que utilizan un curso online conviven diariamente con un teléfono móvil capacitado para correr este tipo de aplicaciones.
- Se adaptará a los diferentes **perfiles de usuarios**. Se tendrá en cuenta qué tipo de alumno está accediendo a *MyLearningMentor*, de forma que, dependiendo de qué tipo de alumno acceda a la aplicación, se le limitará la cantidad de cursos a los que puede apuntarse. Gracias a esto, la aplicación se asegura de que el usuario tiene tiempo suficiente para poder realizar todos los cursos en los que se va matriculando.
- La aplicación facilitará al usuario un **planificador** adaptable. *MyLearningMentor* ofrecerá a cada usuario una lista ordenada de las tareas que debe ir realizando, mostrando al principio aquellas que tienen mayor importancia y una fecha de entrega más cercana en el tiempo.
- La aplicación ofrecerá a los usuarios la posibilidad de acceder a una serie de **consejos** que les ayuden a avanzar en su día a día y a completar los cursos.
- Se obtendrá **información** del tiempo que cada usuario ha tardado en realizar cada tarea. Con esto se quiere conseguir una fuente de información para que en un futuro se pueda mejorar el planificador.

3.1.2 Arquitectura

Se partirá de la arquitectura inicial planteada en el capítulo 2, y se realizarán diferentes modificaciones para adaptarse a los requisitos planteados en el punto anterior. La nueva arquitectura de *MyLearningMentor* que se presenta en esta sección será la posteriormente implementada.

En el modelo inicial se hacía referencia a tres bases de datos de información, una para los perfiles de usuario, otra para la información de los cursos y otra para la retroalimentación de información. Las tres bases de datos serán agrupadas en una sola base de datos. Dentro de la base de datos existirán diferentes tablas. De forma que, cada una de las bases de datos planteadas en el capítulo 2, se transformarán ahora en tres tablas dentro de la misma base de datos. Además de las tablas ya mencionadas se crearán hasta tres tablas de almacenamiento de información más. Una de las nuevas tablas es simplemente una extensión de las tablas ya mencionadas mientras que otras son totalmente nuevas pero necesarias para el desarrollo de la aplicación. Estas nuevas tablas serán explicadas a lo largo de la sección.

Se ha visto la necesidad de incluir dos sistemas de bases de datos. Se incluirá tanto un sistema de bases de datos relacionales como uno de bases de datos documentales. Esta necesidad ha venido motivada principalmente por querer realizar una aplicación preparada para una gran carga de usuarios, de cursos y de tareas. Con las bases de datos documentales se gana gran capacidad y velocidad en las búsquedas que no son perfectamente indexadas, es decir, no tienen un identificador único claro.

Para poder realizar una base de datos indexada es necesario que cada una de las filas que se almacenen en la base de datos tenga un identificador único. En el caso de *MyLearningMentor* sería realmente complicado guardar en una misma tabla todas las tareas de todos los cursos con un identificador único ya que habría que almacenar una cantidad de filas muy elevada. Por ejemplo muchos cursos tienen más de un centenar de tareas, incluyendo los recursos de aprendizaje (por ejemplo vídeos) y las actividades de evaluación (por ejemplo ejercicios de corrección automáticas). Una tabla con un número tan elevado de entradas sería muy ineficiente a la hora de buscar las tareas que corresponden a un curso. En la *Ilustración 8*, se muestra la utilidad de las bases de datos documentales frente a las bases de datos relaciones cuando se trata de información no indexada. Por tanto, si al almacenar la información se tiene un identificador único como por ejemplo cuando se añaden usuarios (email) o cursos (URL) se utilizará una base de datos *MySQL*, en caso contrario, se utilizará una base de datos *MongoDB* (tareas).

Por tanto se definirán un total de seis tablas de almacenamiento de información diferentes. Cuatro de ellas se definirán como tablas de una base de datos relacionales y dos de ellas como colecciones de bases de datos documentales.

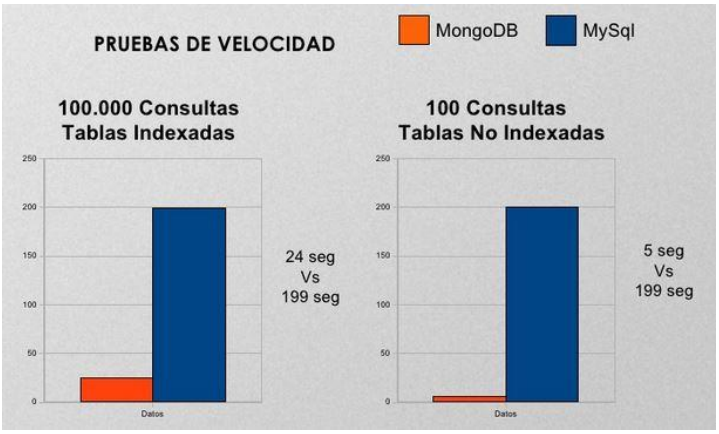


Ilustración 8: Pruebas de velocidad. MongoDB-MySQL (Figura tomada de [28])

Dentro de la base de datos relacional existirá una primera tabla donde se guardará la información del perfil de cada usuario. En la segunda tabla se almacenará la información relativa a los diferentes cursos que se vayan registrando en el sistema. La tercera tabla se utilizará para almacenar la información que establecerá las relaciones entre cada usuario y los cursos a los que esté apuntado. Por último habrá una tabla que contendrá los consejos que la aplicación facilitará a los usuarios.

Dentro de la base de datos documental habrá dos colecciones distintas. La primera de ellas almacenará para cada curso sus tareas, con la información necesaria de cada tarea. La segunda almacenará la relación entre cada usuario y las tareas de cada curso en el que el usuario esté matriculado.

Resumiendo, las tablas y colecciones que tendrá la aplicación quedan agrupadas tal y como se muestra a continuación.

- Tablas de la base de datos relacional:
  - Información de usuario
  - Información de los cursos
  - Relación usuario-cursos
  - Almacenamiento de consejos
- Colecciones de la base de datos documental:
  - Tareas de un curso
  - Relación usuario-curso-tarea

En relación a los **servicios** propuestos en la arquitectura inicial todos serán tenidos en cuenta a la hora de la implementación de la aplicación pero con algún refinamiento en relación a los iniciales. El servicio *Cuenta y gestión de perfiles* que se encarga de gestionar las cuentas y el perfil de los usuarios. La aplicación permitirá a los usuarios crear una cuenta con la que más tarde podrán acceder a *MyLearningMentor*. Una vez que un usuario haya accedido a la aplicación tendrá dos opciones relacionadas con su información personal. Por una parte, se encuentra la información relacionada con sus datos personales donde podrá modificar datos tales como nombre, apellidos, edad, contraseña y correo. Por otra parte tendrá el perfil de

estudiante donde podrá introducir (o modificar) los datos relacionados con su perfil de estudiante tales como el nivel de estudios o el tiempo disponible para estudiar a la semana.

El servicio *Directorio de MOOCs* que proporcionará *MyLearningMentor* gestionará la información de los MOOCs almacenados en el sistema. El usuario podrá añadir nuevos cursos o acceder a los que ya están almacenados en el sistema. La forma de acceder a ellos será mediante un buscador en el cual el usuario deberá poner una palabra clave a partir de la cual la aplicación le mostrará todos los cursos que tengan relación con esa palabra clave. Una vez en el curso seleccionado, *MyLearningMentor* ofrecerá al usuario la posibilidad de apuntarse al curso añadiendo una fila en la tabla usuario-cursos de la aplicación (la aplicación no matricula a los usuarios en los cursos de cara a las plataformas de distribución de MOOCs, sólo lo realiza a nivel de la aplicación). El usuario podrá gestionar los cursos a los que está apuntado así como desapuntarse de ellos (siempre a nivel de la aplicación no de cara a las plataformas de distribución de MOOCs).

Actividades sugeridas de los MOOCs es el tercer servicio que proporcionará *MyLearningMentor*. Este servicio se corresponde con la capacidad de ofrecer al usuario un planificador de tareas según los cursos en los que el usuario se haya matriculado. *MyLearningMentor* facilitará al usuario una lista de tareas ordenadas según la importancia y la fecha de entrega de las tareas que tenga pendientes para todos los MOOCs en los que se encuentre matriculado.

El cuarto, y último servicio que se tenía en cuenta en la arquitectura inicial de la aplicación es *Recopilación de información*. Con este servicio *MyLearningMentor* se encargará de añadir a los cursos sus tareas a partir de la información introducida por los usuarios.

Se ha añadido un nuevo servicio en comparación con los expuestos en la arquitectura inicial de la aplicación. El nombre del servicio es *Directorio de tareas.* La funcionalidad de este nuevo servicio, podía haber sido incluida como parte del servicio *Directorio de MOOCs*, pero por su importancia se ha decidido separar como uno independiente. Este servicio permitirá al usuario ver las actividades que cada curso tiene almacenadas pero, aparte de esto, también le permitirá añadir, modificar o borrar las tareas de un curso.

Como resumen de todos los servicios a implementar se obtiene la lista mostrada a continuación.

1. Cuenta y gestión de perfiles.
2. Directorio de MOOCs.
3. Actividades sugeridas de los MOOCs.
4. Recopilación de información.
5. Directorio de tareas.

La mayoría de la información que se va a almacenar en *MyLearningMentor*, además de para ser mostrada en varias fases de la aplicación, se utilizará para crear un planificador adaptativo individual para cada usuario. Éste es el proceso principal que se ejecutará en la aplicación. El planificador forma parte del servicio *Actividades sugeridas de los MOOCs.* El planificador

completa su información con las tareas que el usuario marca como completadas, indicando también el tiempo efectivo que ha tardado en completarlas.

3.1.3 Funcionalidad

En esta sección se van a mostrar ordenadas por bloques todas las funcionalidades que *MyLearningMentor* ofrecerá a los usuarios en sus distintas partes. Estos serán los requisitos funcionales que en el capítulo siguiente serán comprobados para validar la aplicación.

1. Tratamiento de usuarios:
    a. El usuario podrá registrarse en la aplicación. Para ello podrá introducir nombre, apellido, email y contraseña. El usuario tendrá que introducir obligatoriamente un correo electrónico y una contraseña dos veces.
    b. El usuario podrá añadir o cambiar sus datos personales.
    c. El usuario podrá añadir su año de nacimiento.
    d. El usuario podrá añadir sus ajustes de estudio dependiendo del perfil de estudiante que tenga. Para ello podrá añadir el nivel de estudios, las horas disponibles para estudiar, la franja horaria en la que prefiere estudiar y los días de la semana que tiene disponibles para estudiar.
    e. El usuario podrá revisar y modificar en el momento que quiera su perfil de estudiante.
    f. Cualquier usuario podrá acceder al sistema siempre que disponga de un par email-contraseña válido.
2. Cursos:
    a. El usuario podrá añadir un curso siempre que cumpla dos requisitos. El primero es si la suma del tiempo necesario dedicado a ese curso por semana con la suma del tiempo a dedicar a los demás cursos a los que ya esté apuntado el usuario, es menor o igual a las horas que el usuario tiene disponibles a la semana. El segundo es si la URL del curso no está aún registrada en el sistema, es decir, si el curso todavía no está disponible para los usuarios. En el caso de que se cumplan los dos requisitos el usuario podrá añadir el curso, así como la información sobre el mismo: nombre, número total de horas, número de lecciones, horas dedicadas por semana, URL, tema del curso, fecha de inicio y de final del curso.
    b. Cualquier usuario en el sistema podrá buscar cualquier curso que ya esté registrado en el sistema. Esto lo realizará a través de un buscador a partir de una palabra clave.
    c. Si al realizar una búsqueda no hay coincidencias con la palabra clave, el sistema avisará de que todavía no hay ningún curso relacionado con ese tema.
    d. El usuario podrá ver la información almacenada del curso y podrá apuntarse solo si cumple que el tiempo necesario dedicado a ese curso por semana, teniendo en cuenta los demás cursos a los que ya este apuntado el usuario, es menor o igual a las horas que el usuario tiene disponibles a la semana.
    e. El usuario podrá desapuntarse de un curso en el que este matriculado, liberando así carga lectiva de la semana. Esto le permitirá apuntarse a otros cursos.

3. Planificador adaptativo:
    a. El usuario podrá visualizar un planificador en función de las tareas que le queden por realizar. Las tareas estarán ordenadas siguiendo una doble clasificación. Primero se mostrarán las tareas obligatorias y ordenadas según su fecha de entrega. Luego se le mostrarán las tareas recomendadas ordenadas según la fecha de entrega. Y por último se mostrarán las opcionales también ordenadas según la fecha de entrega.
    b. Al presionar sobre una de ellas el usuario podrá ver su información y podrá marcarla como realizada, indicando el tiempo que le ha supuesto terminarla.
    c. El usuario podrá visualizar las tareas ya realizadas.
4. Tareas:
    a. El usuario podrá añadir tareas a los cursos, indicando para cada tarea nombre, duración, fecha de entrega, tipo de actividad, orden y preferencia. Al añadir una tarea, el usuario tiene que completar obligatoriamente el nombre y el orden.
    b. El usuario podrá editar una tarea en cualquier momento.
    c. El usuario podrá eliminar una tarea en cualquier momento.
    d. El sistema controlará las tareas de forma que no podrá haber dos tareas que compartan el mismo orden, pero siempre se respetará el orden de la última tarea introducida. El resto de las tareas modificarán su orden dependiendo del orden introducido en la última tarea.
    e. Cuando el usuario no pone fecha de entrega en la tarea el sistema cogerá automáticamente la fecha de final del curso al que pertenezca la tarea.
5. Realimentación:
    a. El sistema guardará el tiempo que el usuario tardará en realizar cada tarea.
6. Consejos:
    a. El usuario podrá acceder a un consejo en el momento que quiera. Los consejos estarán precargados en la base de datos correspondiente y se le mostrará uno de forma aleatoria.

3.1.3 Diseño de las bases de datos

Como se ha comentado anteriormente, la aplicación usará hasta seis lugares diferentes para almacenar la información. Siendo cuatro de ellos tablas de una base de datos relacional y dos de ellos colecciones de una base de datos documental. En esta sección se analizará cada una de ellas y se explicarán campo a campo.

Se empezará con las tablas de la base de datos relacional. Como ya se ha explicado en el capítulo 2, el SGDB elegido para dichas bases de datos es *MySQL*. A continuación se expondrán las cuatro tablas de la base de datos relacional (*Profiles, Tips, Mooc_Data y User_course*). Después se desglosarán las colecciones de la base de datos documental (*TaskCourse* y *Course_user*) que como ya se comentó en el capítulo 2 se implementará mediante *MongoDB*. La base de datos contará con dos colecciones, dentro de cada cual se guardarán los diferentes documentos donde residirá la información. Es interesante apuntar este cambio de nomenclatura, según la cual en *MongoDB* se habla de colecciones cuando en *MySQL* se habla de tablas. Los documentos serán las diferentes entradas que tenga una colección.

Profiles

En la *tabla 3* se muestra la estructura de la tabla *Profiles.* En la tabla *Profiles* se guardará la información exclusiva de cada usuario, a la cual sólo el usuario identificado con "email" y "password" correctos tendrá acceso. Cabe destacar aspectos importantes de esta tabla como que el campo de la contraseña se guarda encriptado con *SHA256* y que "email" es un campo único (*primary_key*), de forma que el sistema no permitirá que dos usuarios tengan el mismo "email". Todos los campos de la tabla son *Varchar* para facilitar la comunicación por JSON.

Los diferentes campos se refieren a la información personal del usuario. Los campos se utilizan para almacenar los siguientes datos (siguiendo el orden mostrado en la *Tabla 3*): nombre, año de nacimiento, email (identificador único del usuario), contraseña, nivel de estudios, tiempo disponible para estudiar cada semana, periodo de tiempo disponible (mañana, tarde o noche) y los días de la semana con tiempo para estudiar. Como ya se ha comentado en el apartado de "funcionalidad" no todos estos campos son exigidos al usuario y podrá modificarlos en cualquier momento desde la aplicación.

CAMPO	TIPO DE CAMPO
Name	VARCHAR(20)
Surname	VARCHAR(20)
YearOfBirth	VARCHAR(10)
Email	VARCHAR(150)
Pass	VARCHAR(160)
Levelestud	VARCHAR(20)
Available	VARCHAR(50)
TimeToStudy	VARCHAR(50)
Monday	VARCHAR(3)
Tuesday	VARCHAR(3)
Wednesday	VARCHAR(3)
Thursday	VARCHAR(3)
Friday	VARCHAR(3)
Saturday	VARCHAR(3)
Sunday	VARCHAR(3)

Tabla 3: Campos de la base de datos "Profiles"

Tips

En la tabla "Tips" se almacenan los consejos que la aplicación ofrecerá de forma aleatoria al usuario cada vez que este quiera acceder a uno de ellos. En la *tabla 4* se muestra los campos de la tabla. Cabe destacar en esta tabla que el campo ID es *auto_increment*, es decir, que cada vez que se añada un nuevo consejo automáticamente este campo tendrá como valor un número mayor en una unidad que el ID del último consejo que ya hubiera sido almacenado en la base de datos. En el campo "Text" se almacenará el consejo tal cual se entregará al usuario.

CAMPO	TIPO DE CAMPO
ID	INT(20)
Text	VARCHAR(300)

Tabla 4: Campos de la base de datos "Tips"

Mooc_Data

En la tabla "Mooc_data" se almacenará la información de los diferentes cursos que los usuarios irán incluyendo en el sistema. Como puntos importantes de la tabla destacan: el campo ID, que aumentará automáticamente cada vez que se añada un nuevo curso al tratarse

de un campo *auto-increment*, y el campo *URL_MOOC*, que se trata de un campo *primary_key*, es decir, dos cursos podrán tener todo el resto de la información igual menos la URL que será, por tanto, única para cada curso.

Los usuarios podrán modificar la información de esta tabla solamente en el momento en el que se cree el curso. Las diferentes campos se referirán a (siguiendo el orden de la *tabla 5*): nombre del curso, número total de horas del curso, número de lecciones del curso, las horas que el usuario debe dedicar al curso cada semana, la fecha de inicio y fin del curso, la URL del curso y la temática del curso.

CAMPO	TIPO DE CAMPO
ID	Int(50)
Course_name	Varchar(50)
TotalHours	Varchar(20)
NumberOfLessons	Varchar(20)
Hours_Spent	Varchar(20)
End	Varchar(20)
Start	Varchar(50)
URL_MOOC	Varchar(90)
Type	Varchar(100)

Tabla 5: Campos de la base de datos "Mooc_Data"

User_course

En la *tabla 6* se muestran los campos de la tabla *User_Course*. En esta tabla se almacenarán internamente en el servidor las relaciones entre los cursos y los usuarios. Es decir, para cada usuario que se apunte en un curso se creará una nueva entrada relacionándolos. Esta tabla será modificada siempre que un usuario se apunte o se desapunte de un curso.

La tabla estará formada por tres campos. El primero de ellos será un identificador, *ID*, que se creará automáticamente al tratarse de un campo *auto_increment*. En el segundo campo se almacenará el usuario (al que se le asociará o desasociará un curso). Para identificar al usuario se almacenará el email ya que es el identificador exclusivo para cada alumno. El tercer campo es el que identifica el curso al cual se quiere relacionar al usuario. Para identificar el curso se almacenará la URL, ya que es el único campo exclusivo para cada curso.

CAMPO		TIPO DE CAMPO
ID		INT(20)
USERCOURSE		VARCHAR(150)
URLCOURSE		VARCHAR(150)

Tabla 6: Campos de la base de datos "URL_COURSE"

TaskCourse

En la colección denominada *TaskCourse* se cumplirán dos requisitos. Por una parte se establecerá la relación entre cada nueva tarea introducida y el curso a la cual ésta pertenece. Por otra parte, se almacenará la información exclusiva de cada tarea. Para conseguir esto, cada documento contendrá información exclusiva de un solo curso, y cada curso sólo estará referido por un documento de la colección, es decir, no podrá haber dos documentos que tengan información del mismo curso.

Un documento se creará cada vez que un usuario decida crear un curso. En este momento la aplicación creará un nuevo documento con cero tareas. El usuario modificará esta colección siempre que quiera añadir una nueva tarea en un curso ya definido. El usuario pasará información al sistema sobre el nombre, tipo, duración, fecha de fin, orden y preferencia de la nueva tarea. El servidor se encargará de agregar cada nueva tarea con un identificador único, que se irá creando dependiendo del número total de tareas que tuviera el curso, a partir de "*task1*". Cada vez que se añada una nueva tarea, el servidor sumará automáticamente una unidad al valor almacenado como "numero de tareas".

Ilustración 9: Documento de la colección TaskCourse

En el caso de que el usuario decida no poner fecha de entrega se le asignará por defecto la fecha del final del curso. Esto es necesario ya que el planificador adaptativo requerirá de esta información para poder ofrecer una planificación adecuada al usuario.

Como resultado de la colección se obtendrán una serie de documentos siguiendo cada uno lo mostrado en la *Ilustración 9.* Cada curso podrá tener un único documento en el que se almacenarán las tareas que vayan siendo introducidas. En la *Ilustración 9* se muestra el ejemplo de un curso con dos tareas registradas.

Course_user

La colección *Course_user* será utilizada para establecer la relación tarea-curso-usuario, de forma que lo que se pretende con esta colección es que la aplicación conozca las tareas ya realizadas por un usuario en concreto. Cada par usuario-curso tendrá un nuevo documento en el cual se definirá la relación usuario-curso-tarea. Para cada tarea quedará almacenado si el usuario ha realizado una tarea ya o todavía la tiene pendiente. Como es lógico, será imposible que haya dos documentos que compartan usuario y curso.

Como en el caso anterior, el documento será creado cuando el usuario se apunte en un curso (bien porque lo haya creado o porque se haya matriculado en alguno ya creado). Este documento se modificará cada vez que **cualquier usuario** añada en un curso una nueva tarea o bien cuando el usuario marque como realizada una tarea. En el primer caso, se actualizará la colección añadiendo una nueva tarea que contendrá el campo 'done' con valor 0 y en el segundo caso actualizará solo el valor del campo *done* correspondiente a la tarea con el valor

1. Como es lógico, y según el número de tareas completadas o no completadas, los valores de los campos 'tareas pendientes' y 'tareas completadas' irán modificando su valor. Como resultado de esta colección se tendrán documentos según la estructura de la *Ilustración 10*, habiendo por tanto tantos identificadores de tareas como tareas tenga el curso.

Ilustración 10: Documento de la colección Course-user-task

3.2 Implementación

La implementación de la aplicación se llevará a cabo, como ya se ha comentado en el capítulo 2, utilizando las tecnologías mostradas en la *tabla 7*. Se va a implementar una aplicación para un dispositivo móvil Android que siga el esquema mostrado en la *Ilustración 11.* Para ello será necesario desarrollar en paralelo la parte de cliente y de servidor necesaria para que se puedan realizar las diferentes funcionalidades.

En la *Ilustración 11* se realiza una aproximación a las diferentes opciones planteadas al usuario. Cuando el usuario accede a la aplicación se le muestra la pantalla inicial. En la pantalla inicial el usuario elegirá entre introducir un email y una contraseña o acceder a la pantalla de registro. Una vez que el usuario ha accedido a la aplicación se le mostrarán 4 diferentes opciones. Cada opción le permitirá acceder a diferentes pantallas.

La primera opción, *Planificador,* le permite al usuario acceder a su planificador personal, elaborado a partir de las tareas pendientes. Dentro del planificador podrá visualizar alguna tarea o acceder a las tareas que ya haya completado. La segunda opción, *Consejos,* le permite al usuario obtener un consejo aleatorio. La tercera opción, *Ajustes,* es la que permite a un usuario acceder y modificar sus ajustes personales y su perfil de estudiante. La última opción, *Cursos,* ofrece al usuario una lista con todos los cursos en los que está matriculado. En cualquiera de los cursos que está matriculado el usuario podrá ver las tareas que tiene y añadir, editar o eliminar alguna tarea. Además el usuario podrá desapuntarse del curso. Finalmente en la opción, *Cursos,* podrá buscar algún curso o añadir uno nuevo. Cuando el usuario busca un curso puede visualizar la información del curso y registrarse en el caso que tenga horas disponibles a la semana.

Sistema operativo	Android
Lenguaje de servidor	PHP
Lenguaje de comunicación	JSON
Base de datos estructuradas	MySQL
Base de datos documentales	MongoDB

Tabla 7: Tecnologías utilizadas

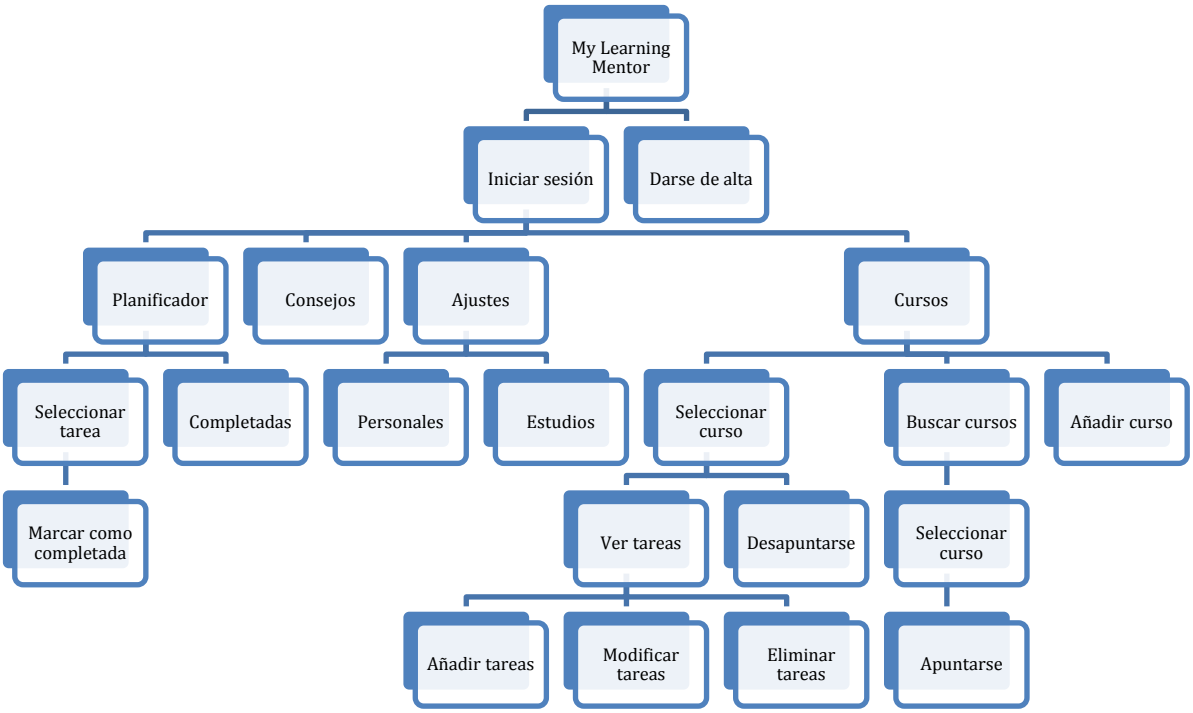


Ilustración 11: Esquema de funcionalidades de MyLearningMentor

3.2.1 Servidor

En la parte de servidor se han desarrollado los documentos PHP necesarios. En total son 25 documentos PHP con los cuales el cliente se conecta dependiendo de las opciones que el usuario vaya seleccionando. Los documentos que ha sido necesarios desarrollar para el correcto funcionamiento de la aplicación quedan indicados, junto a una pequeña explicación de la función que cumplen, en la *tabla 8*.

Excluyendo los archivos auxiliares, que no tendrán conexión con ninguna base de datos, todos los demás documentos seguirán una estructura parecida: recibirán la información necesaria del cliente por medio de JSON, realizarán las tareas necesarias, y devolverán al cliente una respuesta también por medio de JSON. Se va a explicar, a modo de ejemplo, alguno de estos documentos en secciones posteriores.

Archivo (.php)	Descripción
Access	Archivo que devuelve si el par usuario-clave es correcto.
AddCourse1	Archivo que permite a un usuario añadir un curso al sistema y que automáticamente le registra en el curso.
AddCourse2	Registra a un usuario en un curso que ya está almacenado en el sistema.
AddUser	Añade un nuevo usuario al sistema con la información recogida.
CourseInformation	Devuelve la información sobre un curso concreto.
PersonalInformation	Devuelve la información de un usuario en concreto. Añade o modifica la información personal.
StudiesInformation	Devuelve la información del perfil de estudio de un usuario.
AddStudiesInformation	Añade o modifica la información del perfil de estudio de un usuario.
Tips	Devuelve un consejo de forma aleatoria.
QuitCourse	Elimina la relación existente entre un usuario y un curso.
AllMyCourse	Devuelve los cursos de un usuario en concreto.
UpdateStateTask	Actualiza el valor del estado de una tarea (De no realizada a realizada).
DeleteTask	Elimina una tarea de un curso.
TaskInformation	Devuelve la información solicitada de una tarea.
AddTask	Añade una nueva tarea en un curso con la información recibida.
MyTaskComplete	Devuelve las tareas que ya han sido realizadas por un usuario en concreto.
MyTaskPending	Devuelve las tareas que no han sido realizadas por un usuario de los cursos a los que dicho usuario está apuntado.
TaskInformation2	Devuelve la información solicitada de una tarea.
UpdatePersonalSetting	Actualiza o añade los datos personales de un usuario.
BdConnect	Realiza la conexión con la base de datos *MySQL*.
ConfigBD	Contiene la información necesaria de la base de datos *MySQL* a la que se tiene que conectar el servidor.
FunctionsBD	Contiene funciones que serán llamadas relacionadas con la base de datos *MySQL* (I). Por ejemplo: función que almacena la información de un usuario en la base de datos.
FunctionsBD2	Contiene funciones que serán llamadas relacionadas con la base de datos *MySQL* (II). Por ejemplo: función que extrae un consejo de la base de datos.

Tabla 8: Resumen de los archivos del servidor

Login.php

La aplicación le pasará por JSON al servidor el par usuario-clave introducido por un usuario. Se creará un objeto *MySQL* que será el encargado de realizar la comprobación de que el par usuario-clave es correcto. El servidor hará la comprobación de que esta información coincide con la almacenada en la base de datos *MySQL* correspondiente (*Profiles*). Con el resultado, positivo o negativo, de la consulta se creará en JSON una respuesta que será enviada al cliente.

MyTaskPending.php

Este archivo es utilizado para entregar al usuario su planificador personal con las tareas que le quedan por realizar. El cliente sólo le pasará al servidor, por medio de JSON, el usuario que está realizando la petición. Se establecerá la comunicación con la colección adecuada de *MongoDB* (*TaskUser*) que será la encargada de entregar las tareas pendientes. Gracias a este archivo se le entregará al cliente el número de tareas pendientes, las tareas ya ordenadas, por *deadline* y por tipo de importancia, y la información de cada tarea necesaria para que la aplicación muestre al usuario la información de una forma clara. Para poder entregar todo esto de forma correcta se harán las peticiones a la base de datos de una forma ordenada. Primero se compararán las tareas según el tipo de importancia. Después todas las tareas del mismo tipo se irán almacenando en formato JSON según la fecha de entrega del usuario.

AddTask.php

Este archivo recibirá la información de una nueva tarea y se encargará de añadir una tarea en un curso. Para ello modificará las dos colecciones de la base de datos de *MongoDB*, realizará la conexión con la colección *TaskCourse* y añadirá en un curso ya existente la tarea. Añadirá la nueva tarea con un identificador único que creará a partir de las tareas que ya estén almacenadas, es decir, en el caso de que el curso ya tenga almacenadas *i* tareas el nuevo identificador será *task(i+1)*. Al añadir una tarea también se tiene que tener en cuenta el orden de la nueva tarea, de forma que el sistema verá cuál es la posición de la nueva tarea en la lista de tareas. Si se trata de una tarea intermedia, el orden de las tareas que fueran por encima en la lista será cambiado por un el orden inmediatamente superior (se le sumará una unidad), con esto se consigue que no existan dos tareas que tengan el mismo orden. Finalmente, se establecerá conexión con la colección *Course_user* para añadir la relación existente entre la nueva tarea y los usuarios que están apuntados al curso al que pertenece la nueva tarea.

DeleteTask.php

El cliente conectará con este archivo cuando el usuario haya decidido eliminar una tarea del sistema. Solamente se creará el objeto necesario para la conexión con *MongoDB,* ya que, no necesita conectarse con la base de datos de *MySQL*. El sistema modificará ambas colecciones. En la colección *TaskCourse* modificará el contenido que tiene la tarea, de forma que toda la información desaparece pero queda almacenado el identificador de la tarea con el valor "*Delete";* con esto se consigue constancia de que esa tarea ha existido pero alguien la eliminado. La colección *CourseUser* será modificada para todos los alumnos, matriculados en el curso al que pertenezca la tarea, actualizando el valor del campo *Done* sea cual sea a "*Delete".*

3.2.2 Conexión servidor-sistemas de almacenamiento de datos

Para poder realizar las diferentes funcionalidades que *MyLearningMentor* ofrece al usuario, es necesario que el servidor tenga conexión con los dos sistemas de bases de datos utilizados (*MySQL* y *MongoDB*). En la *tabla 9* se muestra con qué sistema de gestión de base de datos tienen conexión los distintos archivos.

Tipo de archivo	Nombre del archivo(.php)
Conexión con *MySQL*	Access AddCourse1 AddCourse2 AddUser CourseInformation PersonalInformation StudiesInformation AddStudiesInformation Tips AllMyCourses
Conexión con *MongoDB*	UpdateStateTask DeleteTask TaskInformation AddTask MyTaskComplete MyTaskPending TaskInformation2
Conexión con *MySQL* y *MongoDB*	UpdatePersonalSetting AddCourse1 AddCourse2 QuitCourse

Tabla 9: Conexión entre servidor y sistemas de almacenamiento de datos

3.2.3 Cliente

Como ya se ha comentado, el cliente ha sido desarrollado para el sistema operativo Android. Se ha creado un proyecto llamado *MyLearningMentor.* Dentro de este proyecto se han creado dos paquetes diferentes. En uno estarán todas las *actividades* que conforman la aplicación con sus respectivas partes como clases o *layouts*. En el otro paquete se ha utilizado una clase ya existente (*Httppostaux.java*), para facilitar las conexiones del cliente Android con el servidor [38].

En la *tabla 10* se hace un resumen de todas las *actividades* existentes en el proyecto. Se mostrará el nombre con una pequeña explicación y los archivos con los que se puede conectar del servidor para cada una de las actividades. El cliente contará con un total de 20 actividades diferentes. En cada una de estas actividades se le ofrecerá al usuario la posibilidad de interaccionar con la aplicación. Con el fin de conseguir una cómoda navegación entre las actividades, se le ofrecerá siempre al usuario la posibilidad de volver a la pantalla inmediatamente anterior. En las *ilustraciones 12, 13 y 14* se muestran 3 capturas de pantallas de diferentes momentos de la aplicación. Además en el anexo D se incluye una tabla con todas las posibles capturas de pantalla de la aplicación junto a una pequeña descripción y los recursos del servidor accesibles por cada actividad.

En la *Ilustración 12* se muestra la actividad *MyPlanner.* En esta actividad el usuario visualizará el planificador con las diferentes actividades que tiene pendientes. Cuando el usuario seleccione una actividad del planificador se le mostrará la información de esta actividad, tal y como se muestra en la *Ilustración 13*. A partir de la actividad en la que se muestra la información de la tarea el usuario podrá indicar que ha realizado dicha tarea. Además, esta actividad le pedirá al usuario que introduzca el tiempo que ha tardado en realizar la tarea. En la *Ilustración 14* se muestra la actividad *SearchCourse* en la cual el usuario visualizará el resultado de una búsqueda realizada. A partir de esta actividad el usuario podrá seleccionar uno de los cursos para visualizar todos sus detalles. En el caso de que el resultado de la

búsqueda no haya sido el deseado el usuario podrá seleccionar la opción de *Añadir un curso no existente* con la cual accederá a una nueva pantalla donde se le permitirá añadir un curso nuevo. Además, en la actividad de la *Ilustración 14* se le ofrece al usuario la posibilidad de realizar una nueva búsqueda.

Todas las actividades tienen una clase Java principal donde se define los elementos de cada actividad. Además la mayoría de las actividades tendrán clases auxiliares dentro de la clase principal. En las clases auxiliares es en las que se realizan las conexiones con las bases de datos correspondientes.

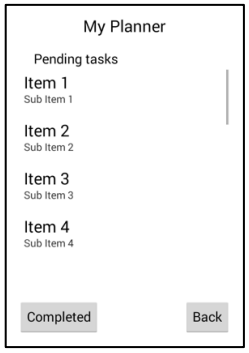


Ilustración 12: Actividad MyPlanner

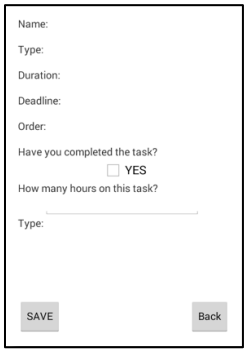


Ilustración 13: Actividad SeeTask

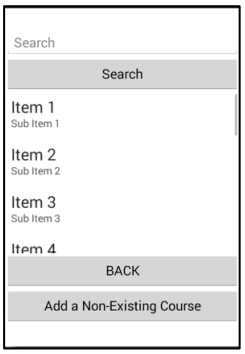


Ilustración 14: Actividad SearchCourse

Nombre	Descripción	Recursos del servidor(.php)
Login	Pantalla para acceder a la aplicación.	Access
AddUser	Introducir los datos para un nuevo usuario.	AddUser
Setting	Opciones de los ajustes.	PersonalInformation, StudiesInformation
Home	Pantalla principal donde el usuario elige qué hacer.	Tips, MyTaskPending, AllMyCourses
Personal	Modificar los datos personales o añadir el año de nacimiento.	UpdatePersonalInformation
Tips	Muestra un consejo de forma aleatoria.	
Studies	Permite añadir los ajustes de estudio.	AddStudiesInformation
MyPlanner	Muestra el planificador adaptativo.	MyTaskComplete, TaskInformation
View Studies	Muestra los ajustes de estudios ya guardados en el sistema.	AddStudiesInformation
SeeTask	Muestra la información de la tarea y permite guardar como realizada la tarea.	MyTaskPending, UpdateStateTask
Complete	Se muestran las tareas ya completadas por el usuario.	MyTaskPending, TaskInformation
SeeTask2	Muestra la información de la tarea cuando accede desde el curso.	DeleteTask, TaskInformation2
SeeTask3	Muestra la información de la tarea desde las tareas completadas.	TaskInformation, MyTaskComplete
AddTask	Sirve para añadir una nueva tarea en un curso.	TaskInformation, AddTask
SearchCourse	Permite poner una palabra para realizar una búsqueda de los cursos.	AllCourses
InforOtherCourse	Muestra la información de un curso al que se ha accedido desde el buscador.	CourseInformation, AddCourse2
Courses	Muestra la lista de los cursos en los que el usuario está registrado.	CourseInformation
OtherCourses	Muestra una lista con los resultados de una búsqueda.	AllCourses
Task	Se muestran las tareas de un curso.	CourseInformation, TaskInformation
AddCourse	Permite añadir la información de un nuevo curso.	AddCourse

Tabla 10: Relación entre cada actividad y los recursos de servidor que requiere

3.2.4 Diagrama de clases

Para la realización del cliente se han realizado una serie de clases. Cada actividad tendrá asociada una clase Java con el mismo nombre de la actividad. Aparte de ésta algunas actividades llevarán clases adicionales encargadas de establecer las conexiones con el servidor. Todas las clases que establecen conexión con el servidor tendrán prefijo *async* y tendrán relación con la clase *Httppostaux,java* (*Ilustración 15*) [38]. En los siguientes diagramas de clases la clase *Httppostaux,java* será omitida para una mejor visualización de los diagramas. Todas estas clases quedan reflejadas en las *Ilustraciones 16, 17, 18, 19, 20 y 21.* En todos los diagramas se muestran todas las relaciones posibles entre las clases. Para cada clase se muestra su nombre, sus atributos y sus métodos. Los diagramas de clases se realizarán a parir del estándar UML [39] y con ayuda de una herramienta compatible con el entorno de desarrollo *Eclipse* llamada *ObjectAid* [40].

Para que el diagrama de clases sea fácil de entender se va a dividir en diferentes figuras. Cada figura abarcará las clases según los requisitos funcionales explicados en la primera parte del capítulo. Según lo expuesto anteriormente existen 6 grupos de requisitos funcionales: *Tratamiento de usuarios, Cursos, Planificador adaptativo, Tareas, Realimentación* y *Consejos.*

En la *Ilustración 16* se muestran las clases necesarias para desarrollar las funcionalidades *Registrarse en el sistema* y *Acceder al sistema* del bloque de requisitos funcionales *Tratamiento de usuarios.* En la *Ilustración 16* también se incluyen las clases para dar servicio al bloque de requisitos funcionales *Consejos.* Además, en la *Ilustración 16* se muestran todas las clases auxiliares de la clase Home. En las siguientes figuras solo se representarán las clases auxiliares de *Home* relacionadas con las funcionalidades representadas en ese diagrama. En la *Ilustración 17* se muestran todas las clases necesarias para cumplir el resto de funcionalidades del bloque *Tratamiento de usuarios (Modificar datos personales, Añadir año de nacimiento, Añadir perfil de estudiante, Editar perfil de estudiante).* En la *Ilustración 18* se muestran las clases necesarias para cumplir las funcionalidades de los bloques de requisitos funcionales *Planificador adaptativo* y *Realimentación*. En la *Ilustración 19* se muestran las clases necesarias para el bloque de requisitos funcionales *Cursos*. En la *Ilustración 20* se pueden observar las clases relacionadas con las funcionalidades *Comprobar orden de tareas, Eliminar tareas* del bloque de requisitos funcionales *Tareas.* Por último, en la *Ilustración 21* se muestran las clases necesarias para los requisitos *Añadir tareas* y *Eliminar taras* del bloque de requisitos *tareas.*

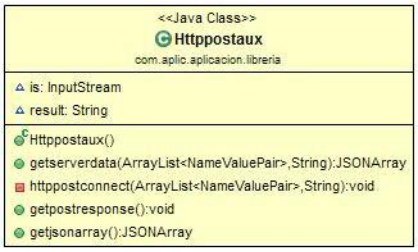


Ilustración 15: Clase Httppostaux

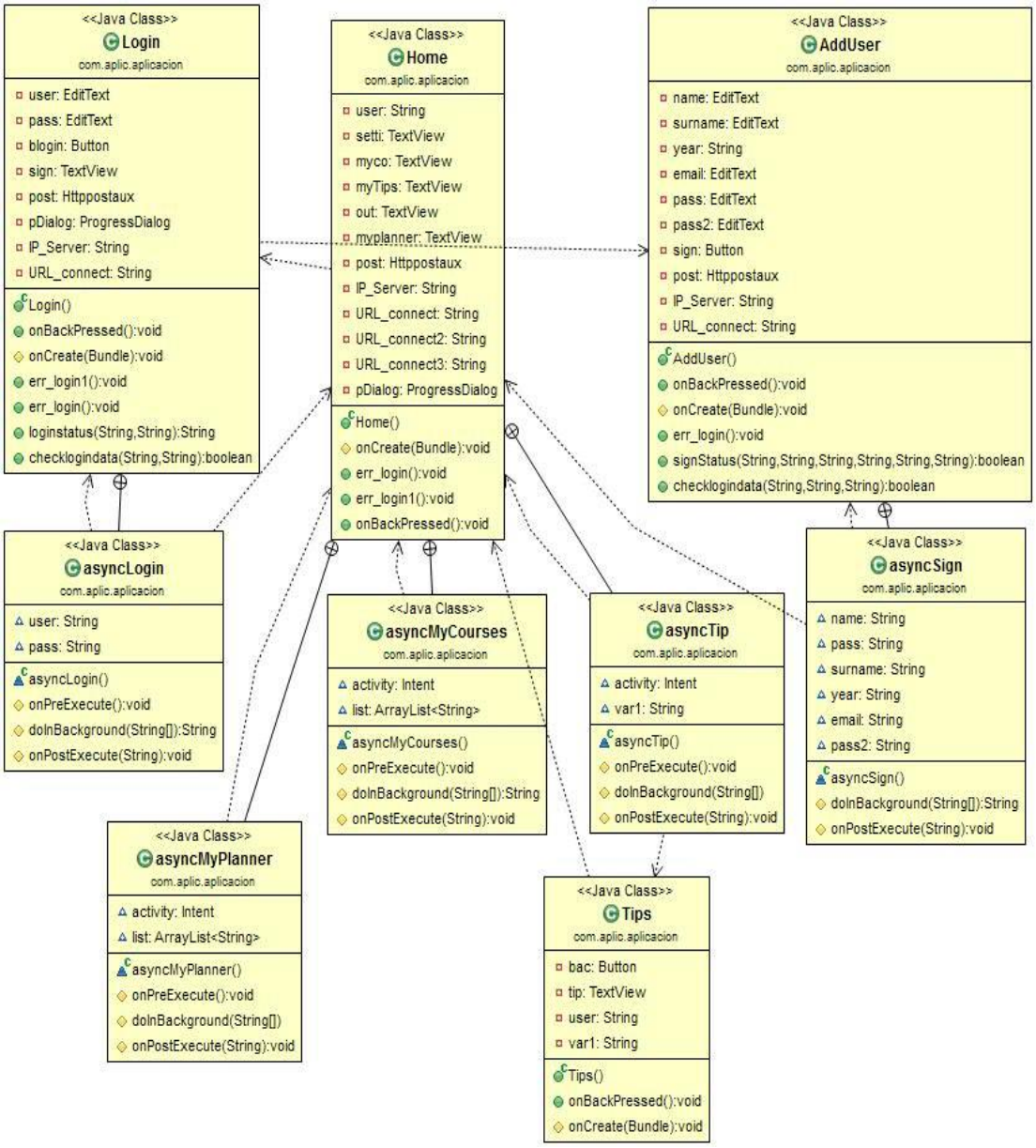


Ilustración 16: Clases del bloque de funcionalidades Tratamiento de usuarios (I) y Consejos

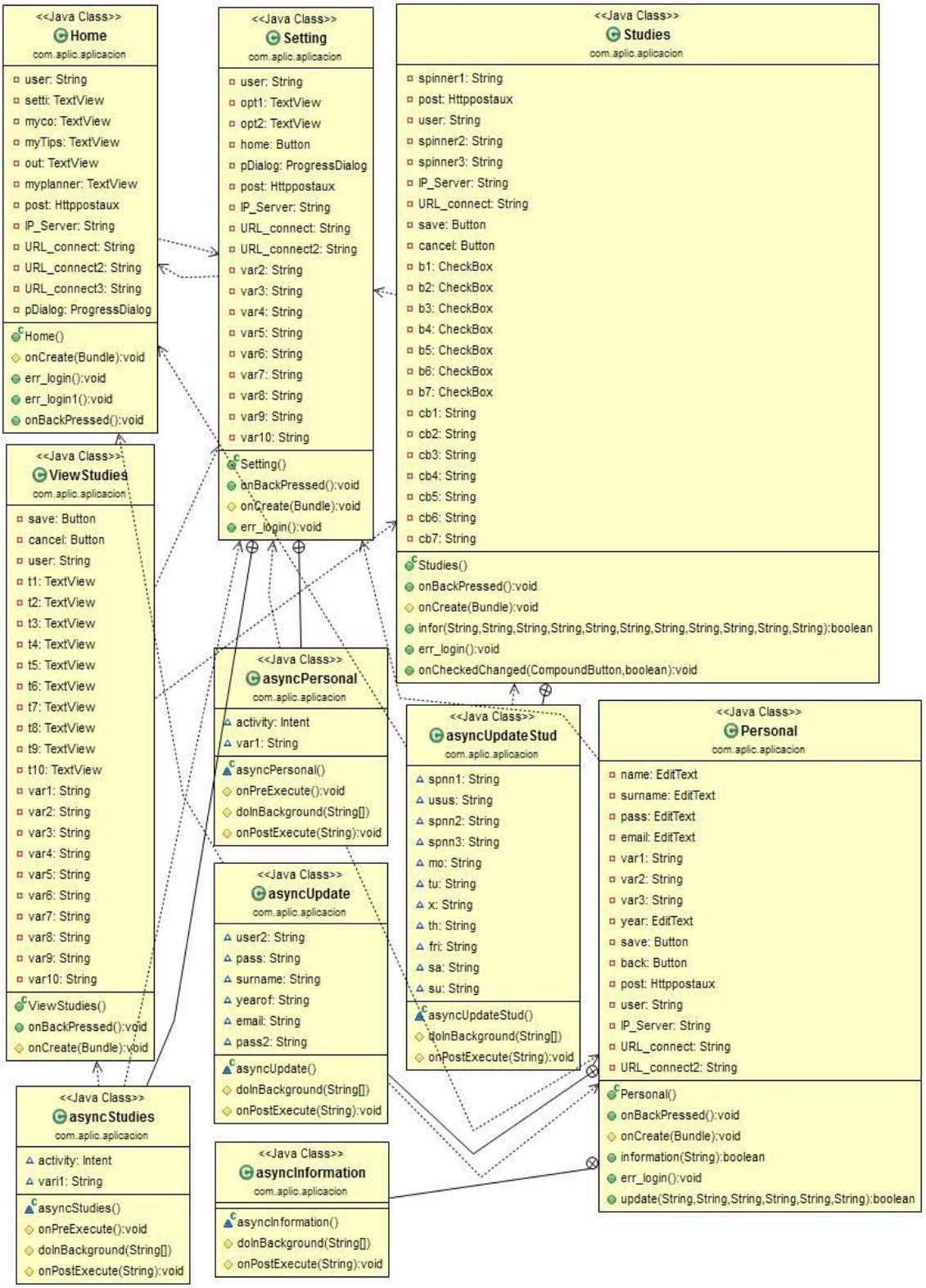


Ilustración 17: Clases del bloque de funcionalidades Tratamiento de usuarios (II)

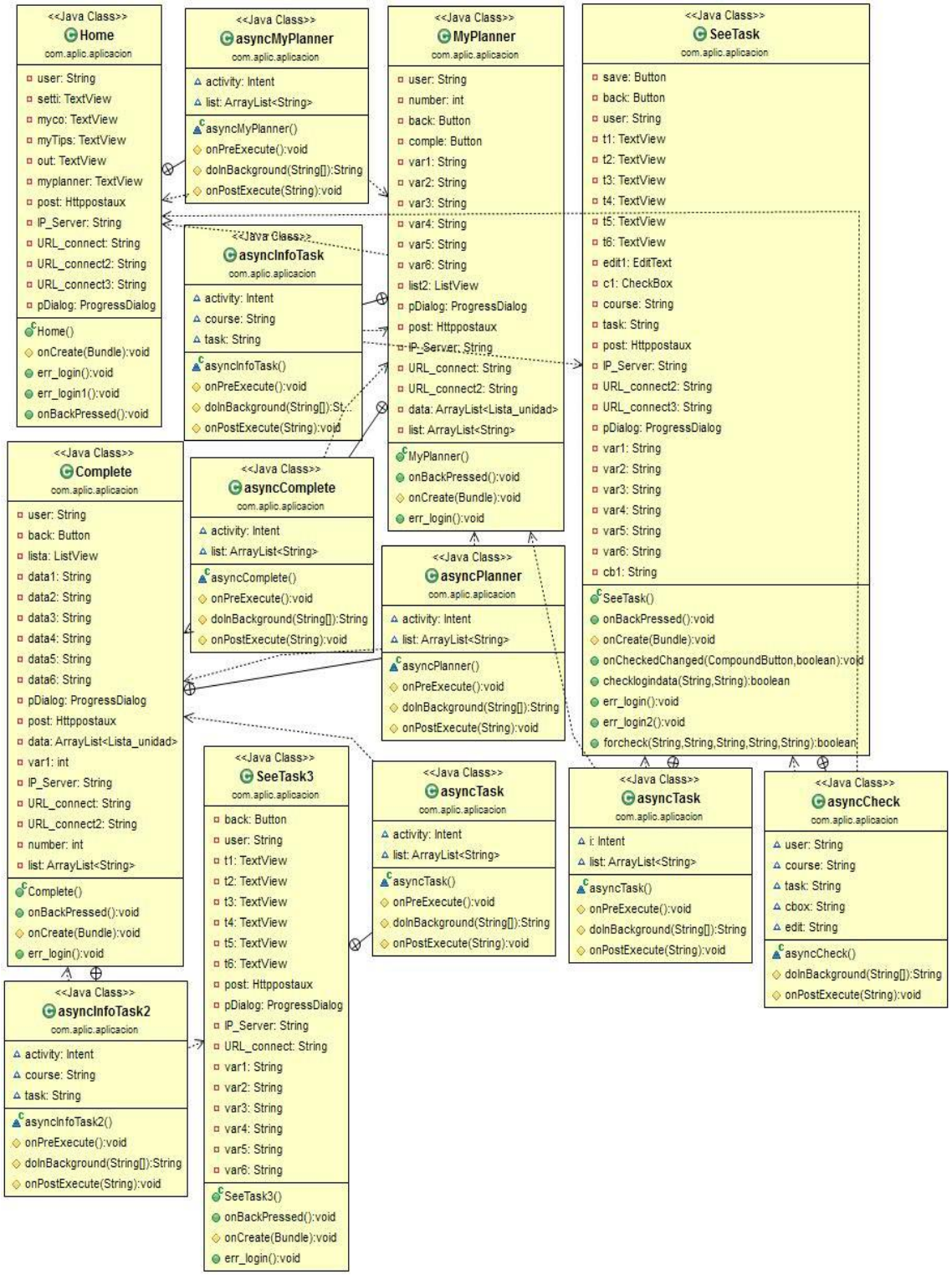


Ilustración 18: Clases para los bloques de funcionalidades Planificador Adaptativo, Consejos y Realimentación

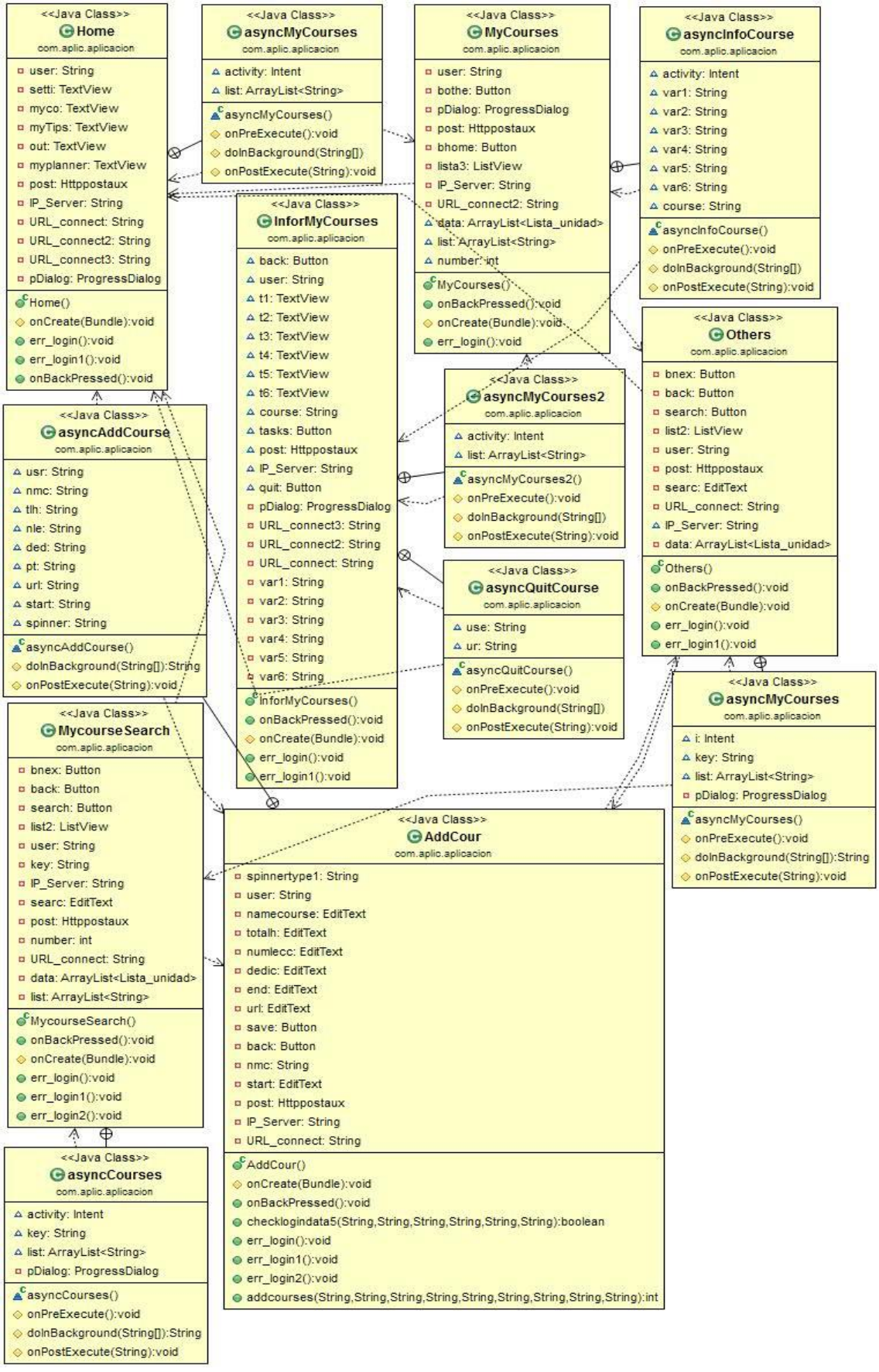


Ilustración 19: Clases para el bloque de funcionalidades Cursos

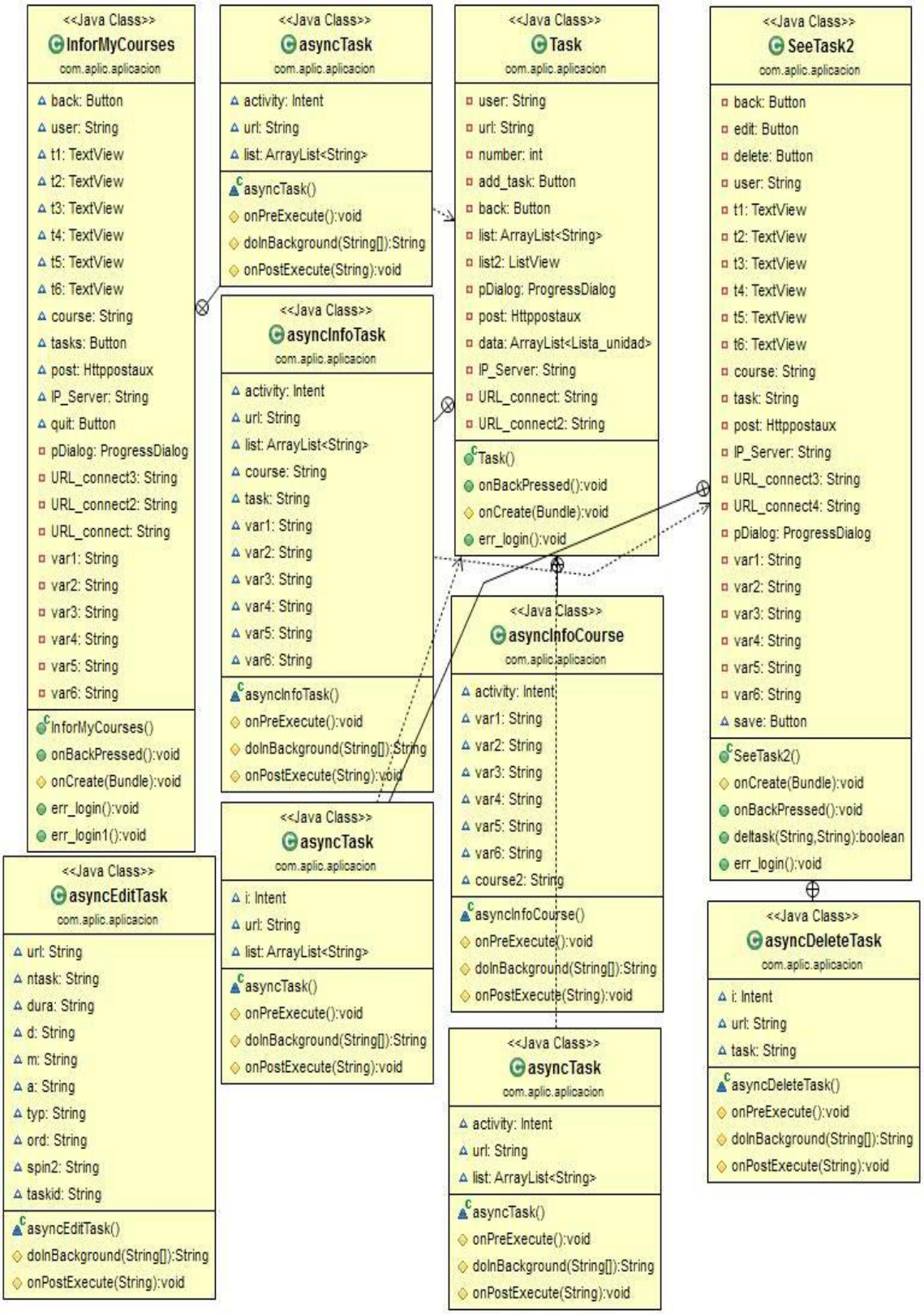


Ilustración 20: Clases del bloque de requisitos tareas

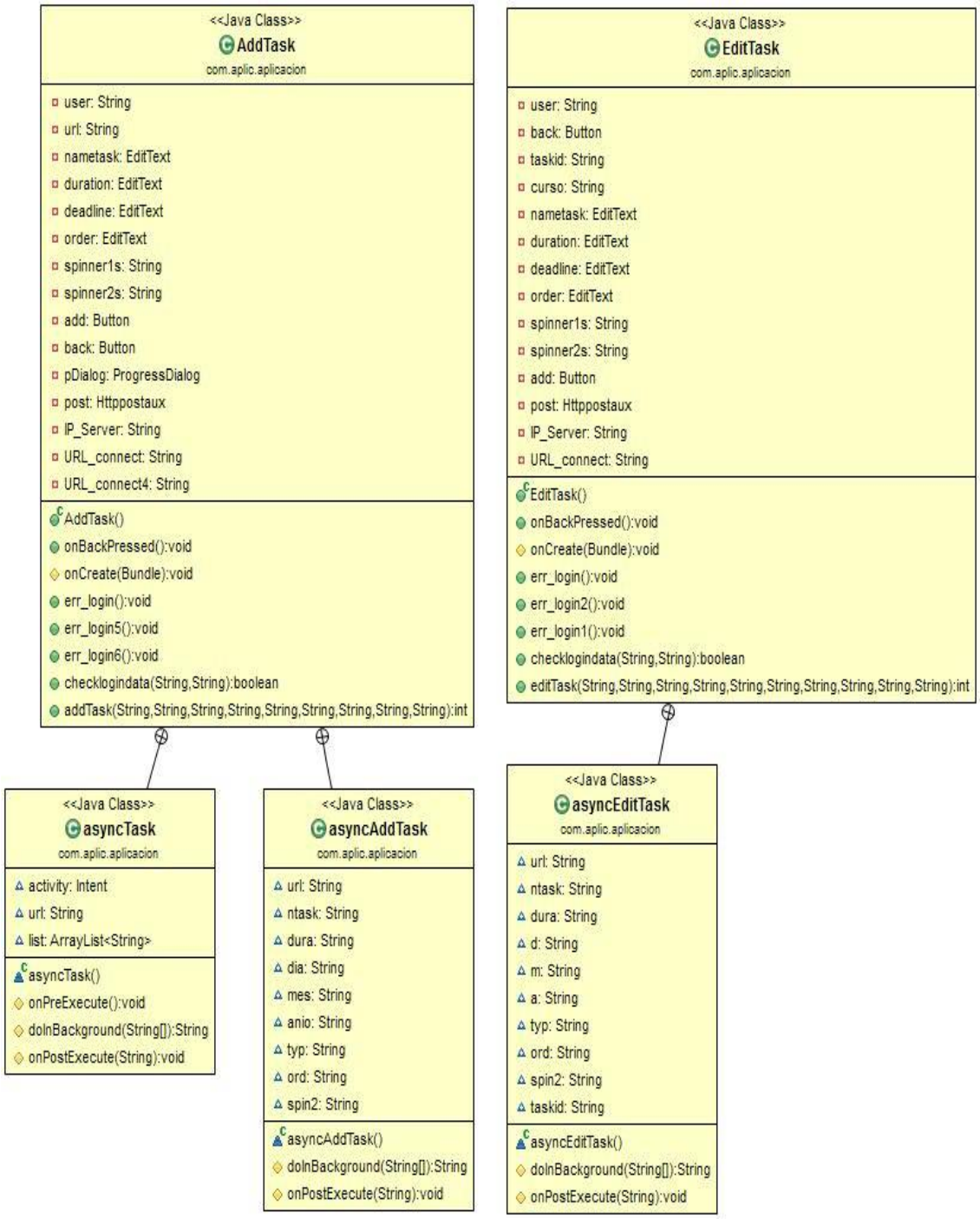


Ilustración 21: Clases del bloque de requisitos tareas (Editar y Añadir tareas)

3.2.5 Diagramas de secuencia

Para entender lo que sucede cuando un usuario interactúa con la aplicación móvil se van a mostrar tres diagramas de secuencia distintos. A partir de estos se entiende que es posible deducir como se realizan las secuencias del resto de actividades. Estos diagramas son mostrados en las *Ilustraciones 22, 23 y 24* donde se muestran los diagramas de secuencia del planificador adaptativo, del listado de MOOC matriculados por el usuario y de la forma de matricularse en un curso ya existente respectivamente. Los diagramas han sido desarrollados siguiendo el estándar UML [39] y utilizando una herramienta online: *Cacoo* [41].

En la *Ilustración 22* se muestra la secuencia de actividades que seguirá la aplicación si un usuario solicita desde el menú principal la opción *Mi Planificador*. El servicio encargado de mostrar las tareas al usuario es *Actividades sugeridas de los MOOCs.* Para poder dar servicio a esta petición la aplicación se conectará con el servidor y con las correspondientes bases de datos para extraer la información necesaria. Lo primero que se realizará será una petición a la colección *CourseUser* de la base de datos de *MongoDB*. De la colección *CourseUser* se extraen los cursos en los cuales el usuario tiene tareas pendientes y el identificador de cada tarea pendiente. Después, el sistema accederá a la colección *TaskCourse* para extraer la información de las tareas a partir de su identificador. Estas dos acciones se repiten tantas veces como en cursos esté matriculado el usuario. Cuando la aplicación tiene la información necesaria de las tareas pendientes del usuario se la muestra al usuario en forma de una lista ordenada. En la lista ofrecida al usuario se podrá seleccionar cualquier tarea para ver más información de ésta.

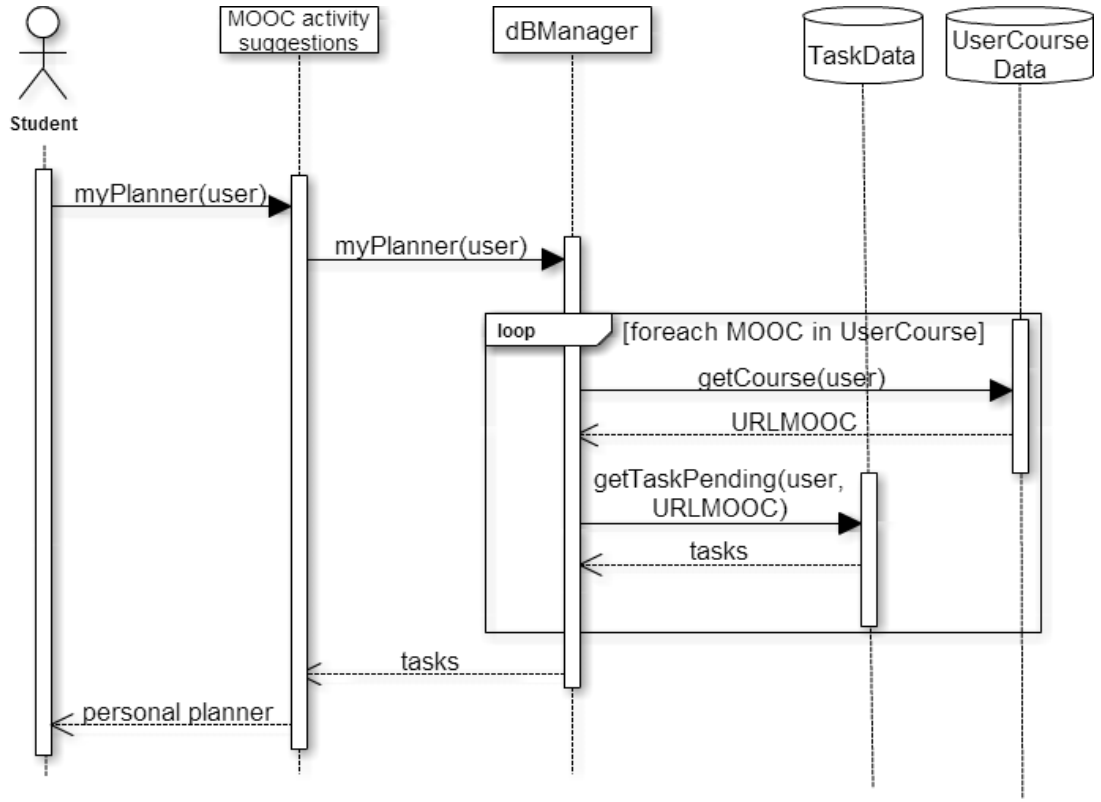


Ilustración 22: Diagrama de secuencia del planificador adaptativo

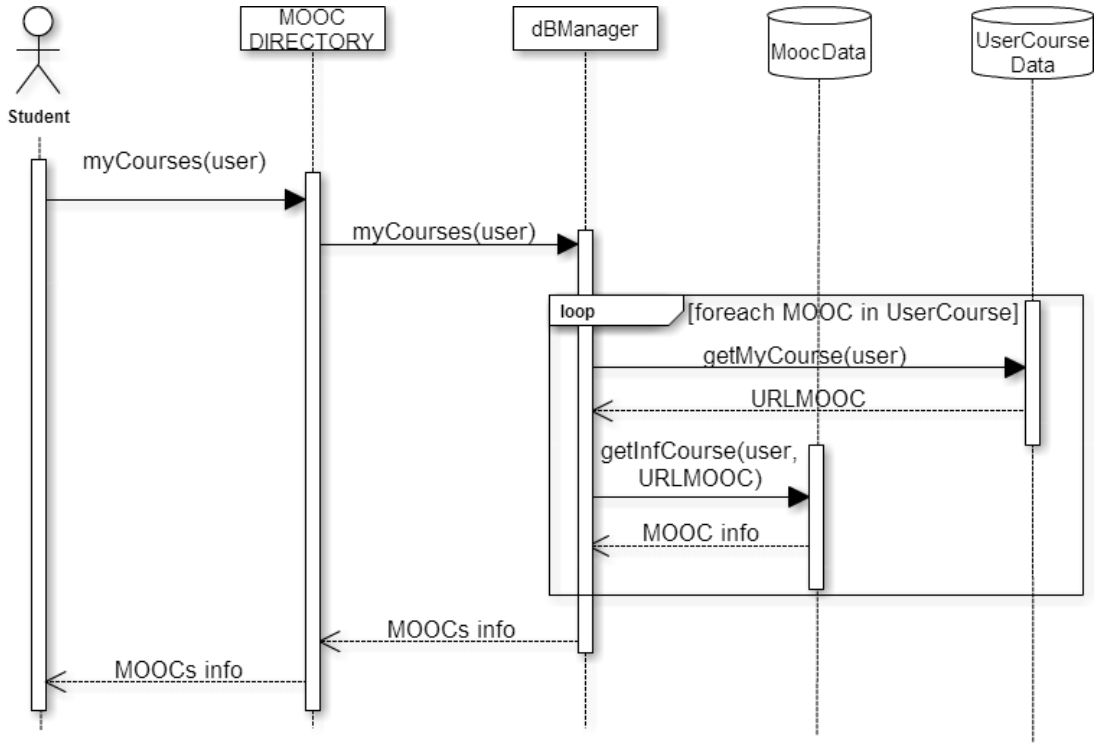


Ilustración 23: Diagrama de secuencia para listar los cursos matriculados por un alumno

En la *Ilustración 23* se muestrá la secuencia de acciones que se realizan cuando el usuario selecciona la opción *Mis cursos* en el menú principal de la aplicación. El servicio implementado que permite listar los cursos matriculados es *Directorio de MOOCs.* La aplicación se conectará con la tabla *Course_user* de la base de datos de *MySQL*, donde se encuentran las relaciones entre cada usuario y los cursos en los que éste está matriculado. Para cada curso en el que el usuario está matriculado el sistema accederá a la tabla *Mooc_Data* para obtener la información necesaria de dicho curso. Esto proceso se repite un número de veces igual al número de cursos en los que el usuario está apuntado. Cuando se tiene toda la información de los cursos matriculados en el servidor preparada, se enviá dicha información a la aplicación. La aplicación mostrará al usuario una lista con todos los cursos en los que está matriculado. En la lista de cursos el usuario podrá seleccionar cualquier de éstos para acceder a la información completa de dicho curso.

En la ilustración 24 se muestra la secuencia de actividades que sigue el sistema cuando un usuario decide matricularse (a nivel de la aplicación móvil) en un curso que ya está almacenado en el sistema. El servicio que hace frente a esta petición es *Directorio de MOOCs.* El usuario buscará el curso a partir de una palabra clave que introducirá en un buscador. La aplicación consultará coincidencias con esa palabra clave en los nombres de los cursos que ya tiene almacenados en la tabla *Mooc_Data.* Las coincidencias (en el caso de que las haya) serán enviadas al usuario. Los cursos se le mostrarán al usuario en forma de lista y el usuario podrá seleccionar un curso de la lista para ver toda la información del curso. En el caso de que el usuario decida apuntarse a algún curso enviará la petición al servidor. Lo primero que se comprobará será si el usuario tiene tiempo disponible en la semana para hacer frente con ciertas garantías de éxito al curso. En caso de que sí que disponga de tiempo, el sistema establecerá conexión con la tabla *User_course* para añadir una nueva entrada con la relación

entre el curso seleccionado y el usuario que ha decidido apuntarse. Por último el servidor enviará a la aplicación una respuesta indicando si ha podido registrar al usuario en el curso o no. En el caso de que no se haya podido registrar al usuario en el curso se le mostrará un mensaje indicándole el motivo. Al matricularse en un nuevo curso el tiempo disponible para matricularse en otros cursos se reduce, pero estos cálculos son realizados la próxima vez que el usuario intente matricularse en un nuevo curso no se almacenará ningún dato con las horas que el usuario tiene disponibles.

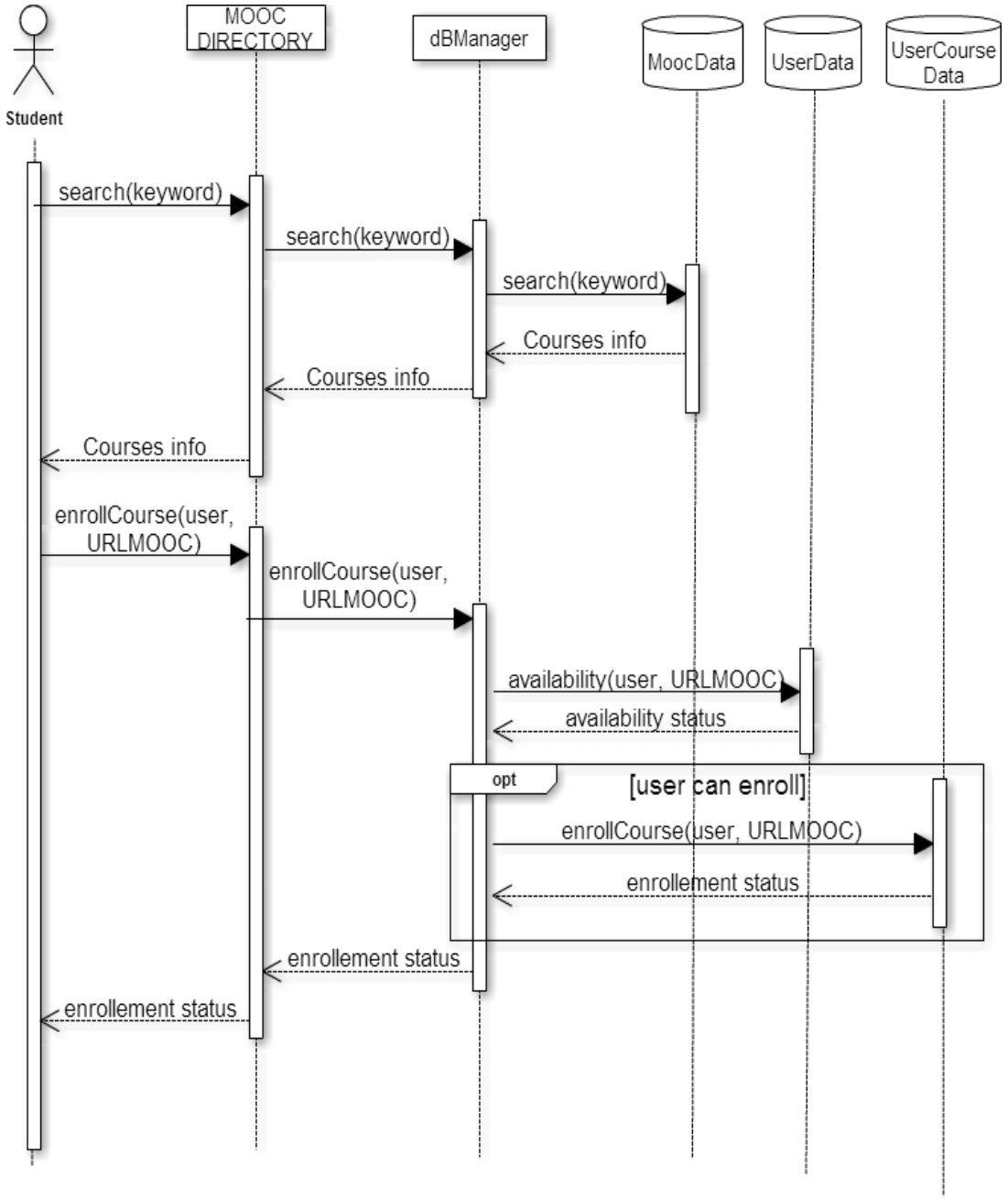


Ilustración 24: Diagrama de secuencia para añadir un curso

3.3 Conclusiones

En este capítulo se ha realizado un refinamiento de la arquitectura inicial de *MyLearningMentor* para llevar a cabo su realización. En la arquitectura final se han definido 5 servicios, 4 tablas de una base de datos relacional, 2 colecciones de una base de datos documental y un proceso. También se han expuesto los requisitos y funcionalidades que debe tener la aplicación a realizar.

En la segunda parte del capítulo se ha realizado la primera implementación de una aplicación siguiendo la arquitectura desarrollada en la primera parte del capítulo. Se ha implementado tanto la parte del cliente como la parte del servidor. En la parte del cliente se han implementado 20 actividades y en la parte del servidor 25 documentos PHP. Para entender el resultado de dicha implementación se han expuesto diferentes diagramas de clases y de secuencia.

Capítulo 4: Validación

En este capítulo se validarán los requisitos funcionales que debe cumplir la aplicación, de acuerdo con lo expuesto en el capítulo 3. Para ello, se realizarán diferentes pruebas con el fin de comprobar que la aplicación desarrollada da soporte a dichos requisitos. Además, en la segunda parte del capítulo se realizarán pruebas de carga y se discutirá cómo se ha abordado el tema de la seguridad en la aplicación.

4.1 Validación de requisitos funcionales

Según lo expuesto en el capítulo 3, la aplicación se ha desarrollado para dar soporte a 6 bloques de requisitos funcionales. Estos bloques son: *Tratamiento de usuarios, Cursos, Planificador Adaptativo, Tareas, Realimentación y Consejos.* Para comprobar que la aplicación realmente cumple con todos los requisitos funcionales se han realizado diferentes pruebas. Para el bloque de requisitos funcionales *Tratamiento de usuarios* se han llevado a cabo dos pruebas: *Registro en la aplicación* y *Actualización del perfil académico.* Para los bloques de requisitos *Cursos* y *Tareas* se han realizado dos pruebas: *Añadir un curso nuevo* y *Matriculación en un curso ya existente.* Para comprobar los requisitos funcionales incluidos en los bloques *Planificador Adaptativo* y *Realimentación* se han realizado dos pruebas: *Manejo del planificador adaptativo* y *Actualización del planificador adaptativo.* Para el bloque de requisitos funcionales *Consejos* se ha realizado una única prueba *Petición de un consejo.* Las diferentes pruebas realizadas para la comprobación de los requisitos serán explicadas a continuación.

En la *tabla 10* se mostrará un resumen con las relaciones entre todas las funcionalidades y las pruebas que se han realizado para comprobar que éstas se cumplen. En la parte izquierda de la *tabla 10* se desglosarán todas las funcionalidades marcadas en el capítulo 3 y en la parte superior las pruebas realizadas en este capítulo. Cada funcionalidad tendrá un ✓ en las pruebas que han servido para su validación.

4.1.1 Registro en la aplicación (Prueba 1)

En esta prueba se va a comprobar que un usuario puede registrarse en la aplicación y actualizar su perfil personal. Un usuario cualquiera decide empezar a utilizar la aplicación *MyLearningMentor*. Al abrir la aplicación se muestra al usuario una primera pantalla donde tiene que insertar un par email-contraseña correcto (*Ilustración 20a*). Como es la primera vez que el usuario accede a la aplicación, no dispone de una cuenta, por lo qué decide registrarse. Para registrarse accede a la pantalla de registro en la cual la aplicación le invitará a introducir sus datos personales para poder realizar el registro (*Ilustración 20b*).

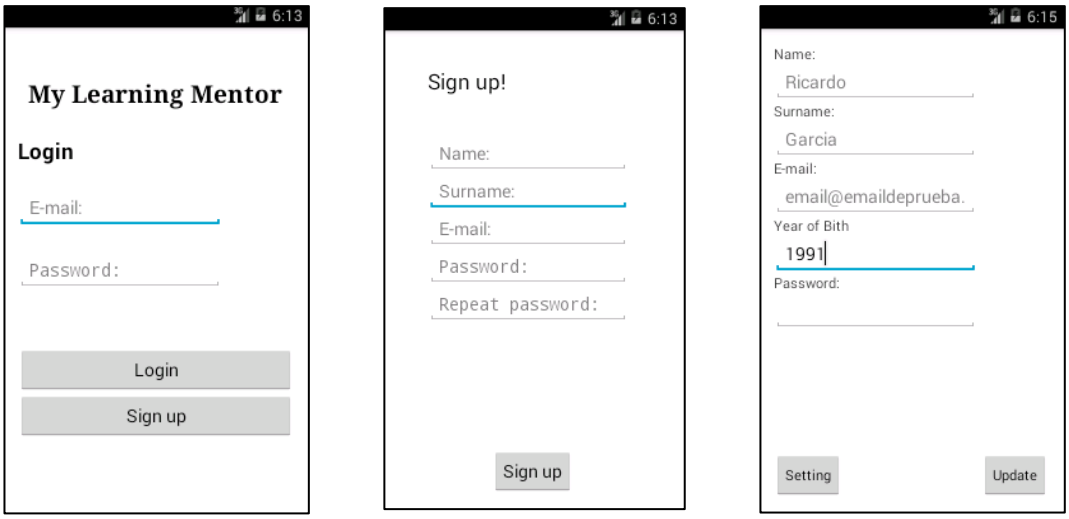


Ilustración 25: Capturas de pantalla prueba Registro en la aplicación. De izquierda a derecha: (a) pantalla principal, (b) pantalla de registro y (c) pantalla de actualización de datos personales

Para poder registrarse el usuario debe introducir un email que no haya sido utilizado previamente en la aplicación. Además debe de introducir dos veces la contraseña elegida. En el caso de que las dos contraseñas no coincidan la aplicación mostrará un mensaje de error y no permitirá el registro. Por último, el usuario decide añadir su año de nacimiento. Para añadir el año de nacimiento el usuario accede a la pantalla de actualización del perfil personal (*Ilustración 20c*). La aplicación comprobará que el año de nacimiento introducido por el usuario sea lógico; en caso contrario mostrará un mensaje de error al usuario pidiendo que introduzca de nuevo este dato.

4.1.2 Actualización del perfil académico (Prueba 2)

En esta prueba se va a comprobar que el usuario puede actualizar el perfil de estudiante. El usuario accederá a la aplicación a través de la pantalla de acceso, introduciendo su email y contraseña (*Ilustración 21a*). Una vez que el usuario haya accedido a la aplicación, seleccionará la opción *Ajustes* del menú principal. En la pantalla ajustes se le ofrecerá la posibilidad de acceder a los datos personales o a sus preferencias de estudio (*Ilustración 21b*). Como en este caso el usuario quiere actualizar su perfil académico, elegirá la segunda opción *Preferencias de estudio.* La aplicación le mostrará una pantalla con sus preferencias de estudio. El usuario seleccionará la opción *Actualizar* y accederá a la pantalla de modificación de sus datos académicos (*Ilustración 21c*).

El usuario podrá modificar su nivel de estudios eligiendo la opción que más se aproxime a su situación entre: educación primaria, educación secundaria, grado, master o doctorado. El segundo dato que tendrá que elegir el usuario es el tiempo que tiene disponible a la semana para dedicar a la realización de las tareas de los MOOCs en los que está matriculado. La aplicación le dejará elegir entre las opciones: <1, 1 a 3, 3 a 5, 5 a 10 o >10. Este dato es importante porque es el dato que marca la cantidad de cursos que la aplicación permitirá al usuario apuntarse. El tercer dato a introducir por el usuario es la franja horaria en la que realizará las tareas de los cursos. El usuario podrá elegir entre: mañana, tarde o noche. Por último, el usuario tendrá que seleccionar los días de la semana en los que prefiere realizar las actividades de los cursos. El usuario tendrá que seleccionar como mínimo un día de la semana; en caso contrario, la aplicación mostrará un mensaje de error al guardar los datos. Antes de

guardar los cambios la aplicación le mostrará una advertencia al usuario indicándole que si acepta los cambios se perderá toda la información que hubiera almacenado previamente. En el caso de que el usuario acepte, su perfil de estudiante será actualizado correctamente.

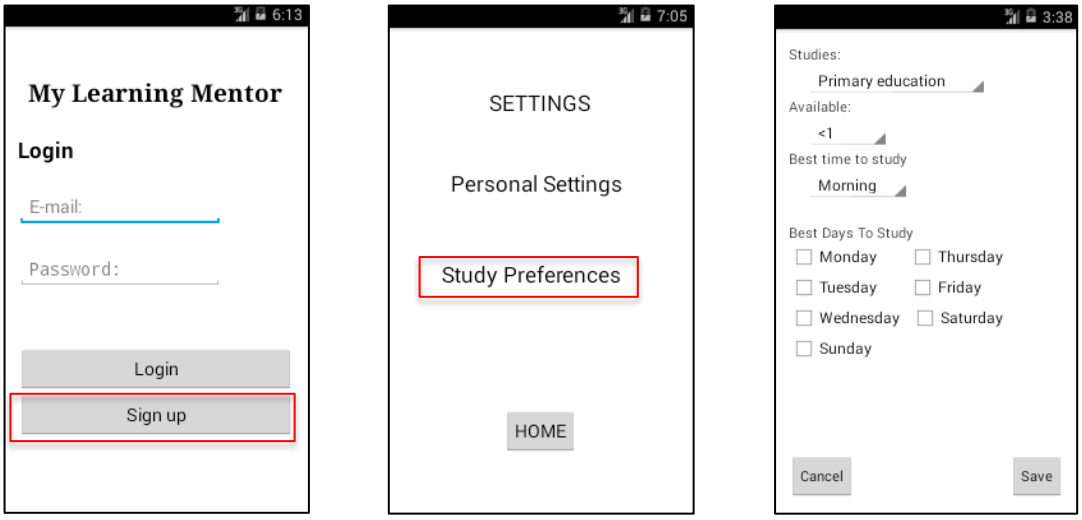


Ilustración 26: Capturas de pantalla prueba Actualización del perfil académico. De izquierda a derecha: (a) pantalla principal, (b) pantalla de ajustes y (c) pantalla del perfil académico

4.1.3 Añadir un nuevo curso (Prueba 3)

En esta prueba se comprobarán las funcionalidades relacionadas con los bloque de requisitos funcionales *Cursos y Tareas*. En concreto se comprobará cómo un usuario puede añadir un curso y añadir, modificar o eliminar tareas.

El usuario accederá a la opción *Mis cursos* del menú principal de la aplicación. En la nueva pantalla se le mostrará al usuario los cursos en los que está matriculado (en el caso de que esté matriculado en alguno). Si el usuario selecciona la opción *Otros* se le mostrarán las opciones que le permitirán realizar una búsqueda de un curso o añadir uno nuevo (*Ilustración 22a*). El usuario antes de ingresar un curso nuevo en el sistema decide buscar el curso. Para buscar un curso introduce una palabra clave y presiona sobre la opción *Buscar.* En el caso de que la búsqueda no haya tenido éxito la aplicación mostrará un mensaje al usuario diciendo que no ha encontrado resultados con esa palabra clave (*Ilustración 22b*). La aplicación también controlará si el usuario presiona sobre *Buscar* pero no ha introducido ninguna palabra avisándole con un mensaje de error de lo sucedido.

Para introducir un curso nuevo, el usuario tiene que introducir la información del curso completa rellenando: nombre del curso, número total de horas, número de lecciones, horas dedicadas por semana, URL del curso, fecha de inicio, fecha de finalización y temática del curso (*Ilustración 22c*). Para la elección de la temática del curso el usuario puede elegir entre un conjunto de temáticas extraídas de la plataforma edX, una de las más populares para ofrecer MOOCs: biología, negocios, química, comunicación, economía y finanzas, electrónica, energía y ciencias de la tierra, ingeniería, estudios medioambientales, alimentos y nutrición, salud y bienestar, historia, humanidades, derecho, literatura, matemáticas, medicina, música, filantropía, filosofía y ética, física, ciencias, ciencias sociales o estadística y análisis de datos. La aplicación también controlará el formato en el que el usuario introduce las fechas de inicio y

finalización del curso. La fechas tendrán que tener el formato *DD-MM-YYYY*, es decir, tendrá que tener dos dígitos para el día, dos dígitos para el mes y cuatro dígitos para el año. Además los valores deben ser valores reales para un día, un mes y año. En caso de que alguna de las fechas introducidas no tengan el formato correcto o que la URL introducida ya haya sido utilizada en el sistema la aplicación mostrará un mensaje de error al usuario.

En la segunda parte de esta prueba el usuario manipulará las tareas correspondientes al curso que acaba de registrar. Para acceder a las tareas del curso seleccionará el curso de la pantalla en la cual se muestran los cursos matriculados (*Ilustración 22d*). En la nueva pantalla que le aparece al usuario seleccionará la opción *Añadir tarea* que le permitirá añadir una nueva tarea al curso seleccionado.

Para añadir una nueva tarea al curso deberá incluir información de la tarea acerca: nombre, tipo, fecha de entrega, duración, orden e importancia. Para indicar el tipo de tarea, el usuario tendrá la oportunidad de elegir entre: video, tarea formativa, tarea complementaria u otros. En el caso de que el usuario no indique la fecha de entrega de la tarea, se le asignará como ésta la fecha de finalización del curso. En el caso de que el usuario sí que haya decidido introducir una fecha de entrega, la aplicación comprobará que la fecha de entrega tiene el formato correcto *DD-MM-YYYY*. El usuario tendrá que indicar obligatoriamente el orden que tiene la tarea en el curso. La aplicación se encargará, según el orden introducido, de modificar el orden del resto de las tareas del curso para que no existan dos tareas con el mismo orden. Para indicar la importancia que tiene la tarea en el curso el usuario podrá elegir entre: obligatoria, recomendada y opcional.

En esta prueba el usuario añadirá dos tareas con orden uno y el resultado se muestra en la *Ilustración 22e.* Donde se puede apreciar que a la tarea que ya estaba almacenada en el sistema se le ha cambiado el orden para que no existan dos tareas con el mismo orden en el mismo curso.

Por último el usuario decide eliminar una tarea. Para llegar a la pantalla donde se encuentra la opción *Borrar* tendrá que acceder primero a la lista de tareas y de ahí seleccionar la que quiera eliminar (*Ilustración 22f*). Al eliminar una tarea quedará una lista ordenada pero no completa (excepto en el caso en el que la tarea eliminada sea la última tarea del curso). La aplicación tendrá esto en cuenta cuando un usuario quiera añadir una nueva tarea y antes de modificar el orden de las tareas almacenadas verificará que entre éstas existe alguna tarea que tenga ese mismo orden, ya que, en caso contrario, no deberá modificar las otras tareas.

Cuando una tarea es eliminada, esta información no es relevante solo para el usuario que ha eliminado la tarea sino que afectará a todos los demás usuarios. Todos los usuarios que tenían en su planificador esta tarea como tarea pendiente o tarea completada verán como la tarea eliminada ha desaparecido.

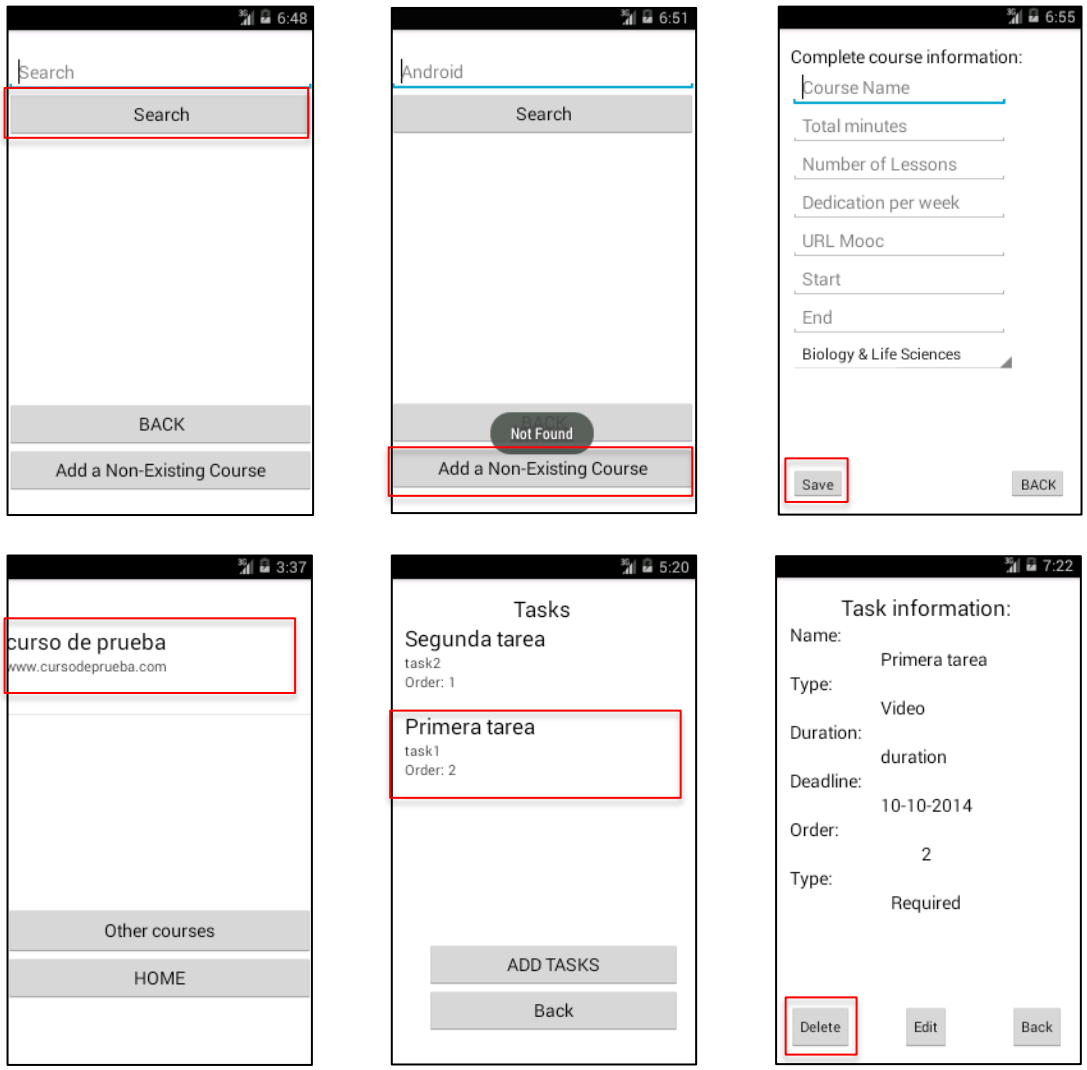


Ilustración 27: Capturas de pantalla prueba Añadir un nuevo curso: De izquierda a derecha y de arriba abajo: (a) pantalla de los cursos matriculados, (b) pantalla después de búsqueda, (c) pantalla de registro de un curso, (d) pantalla con los cursos matriculados, (e) pantalla de las tareas de un curso y (f) pantalla con la información de una tarea

4.1.4 Matriculación en un curso ya existente (Prueba 4)

En esta prueba se va a comprobar la posibilidad de que un usuario se matricule en un curso ya existente. Para ello, el usuario accederá a la pantalla de búsqueda de un curso e introducirá una palabra clave (*Ilustración 23a*). Cuando un usuario realiza una búsqueda con coincidencias podrá seleccionar alguna de éstas para acceder a la información del curso seleccionado. En esta pantalla se le mostrará al usuario la información del curso en relación a: nombre, número total de horas, número de lecciones, número de horas de dedicación a la semana, fecha de finalización del curso y URL el curso (*Ilustración 23b*). En esta pantalla el usuario tendrá la opción de matricularse, siempre que tenga tiempo disponible y no esté matriculado ya (en este caso se le mostrará un mensaje error). Una vez matriculado en el nuevo curso, el usuario accederá a la pantalla de sus cursos matriculados desde el menú inicial para comprobar que realmente está apuntado (*Ilustración 23c*).

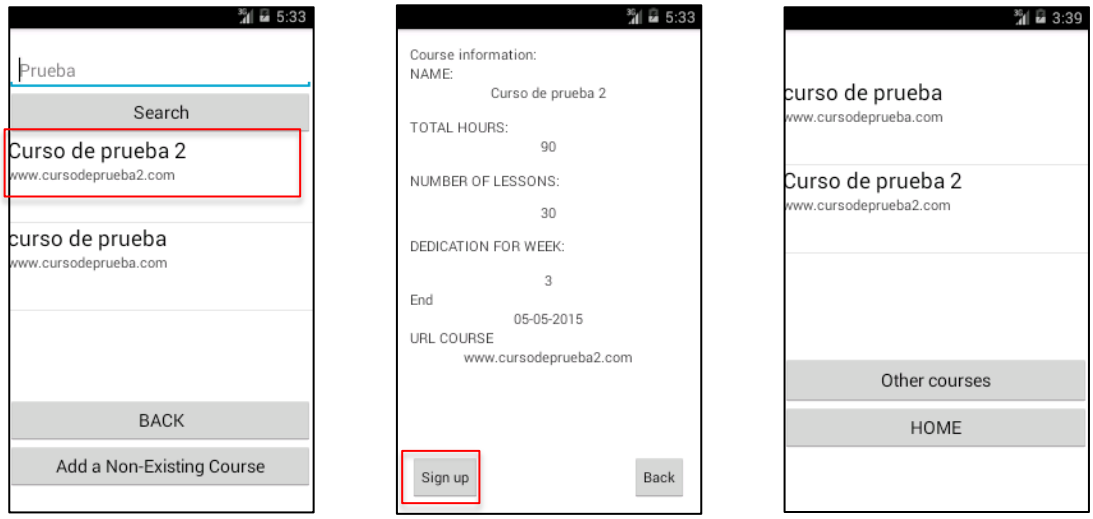


Ilustración 28: Capturas de pantalla prueba Matriculación en un curso ya existente. De izquierda a derecha: (a) pantalla de búsqueda de cursos, (b) pantalla de información de un curso y (c) pantalla de cursos matriculados

4.1.5 Manejo del planificador adaptativo (Prueba 5)

En esta prueba se va a comprobar que el planificador adaptativo funciona correctamente. Para ello se van a comprobar los bloques de requisitos funcionales *Planificador adaptativo* y *Realimentación.* El usuario accede a la aplicación y desde el menú principal seleccionará la opción *Mi planificador (Ilustración 24a)*. La aplicación mostrará el planificador al usuario. El planificador contendrá todas las tareas que tenga pendientes por realizar el usuario. Las tareas estarán ordenadas siguiendo una doble clasificación: según importancia y según fecha de entrega de cada tarea. El usuario podrá seleccionar una de las tareas mostradas por el planificador para acceder a la información de la tarea. En la nueva pantalla se le mostrará la información de la tarea además de darle la opción de marcar como realizada la tarea seleccionada. Cuando el usuario indica que ha realizado la tarea, debe indicar también el tiempo en minutos que la ha supuesto el realizarla (*Ilustración 24b*). En el caso de que el usuario intente marcar una tarea como realizada pero no indique el número de minutos, que ha usado para realizarla, la aplicación le mostrará un mensaje de error recordándole que debe rellenar el dato. En esta prueba el usuario marcará como realizada una tarea y accederá a ver las tareas que ha realizado. Para ver las tareas que ha realizado, debe acceder al planificador y seleccionar la opción *Tareas completas.* En esta pantalla la aplicación mostrará al usuario las tareas que ya haya realizado de los cursos en los que actualmente esté matriculado (*Ilustración 24c*). Las tareas realizadas de cualquier curso en el que el usuario ya no esté matriculado no se le mostrarán para que el usuario se centre exclusivamente en los cursos matriculados. Además, el usuario en la lista de tareas completadas puede seleccionar una de estas tareas para ver la información de la tarea que ya ha realizado.

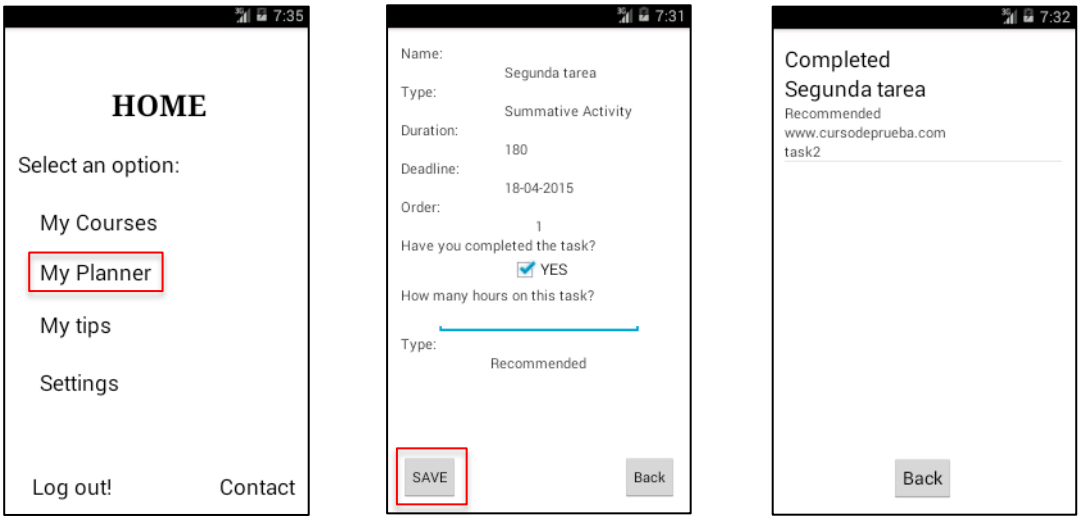


Ilustración 24: Capturas de pantalla pruebas Manejo del planificador adaptativo y realimentación. De izquierda a derecha: (a) pantalla del menú principal, (b) pantalla de tarea seleccionada, (c) pantalla de tareas completadas

4.1.6 Actualización del planificador adaptativo (Prueba 6)

En esta prueba se comprobarán parte de las funcionalidades de 2 de los 6 bloques de funcionalidades: *Cursos* y *Planificador adaptativo.* Lo primero que hará el usuario será acceder desde el menú principal a la opción *Mis cursos*. En la nueva pantalla seleccionará el curso del que quiera borrarse y la aplicación le mostrará la información del curso con la opción *Quitar curso* (*Ilustración 25a*). Cuando el usuario seleccione la opción *Quitar curso* se eliminará ese curso de la base de datos correspondiente al usuario y se actualizará el planificador (las tareas del curso seleccionado, tanto pendientes como completadas, serán eliminadas). Por último, desde el menú principal el usuario elegirá la opción *Mi planificador* para comprobar que todo se ha actualizado correctamente (*Ilustración 25b*).

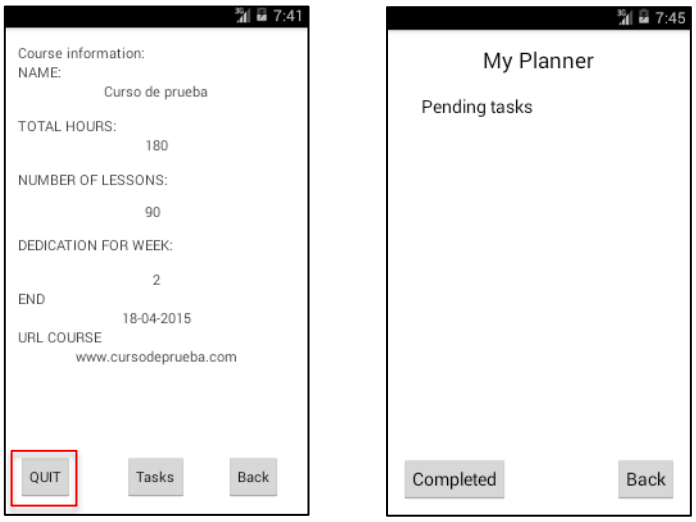


Ilustración 25: Capturas de pantalla prueba Actualización del planificador adaptativo. De izquierda a derecha: (a) pantalla de información de mis cursos, (b) pantalla del planificador

4.1.7 Petición de un consejo (Prueba 7)

En esta prueba se va a realizar la comprobación de que un usuario pueda pedir un consejo. Para realizar esta prueba el usuario accederá a la opción *Consejos,* desde el menú principal de la aplicación. Al seleccionar la opción *Consejos* la aplicación le mostrará al usuario una nueva pantalla con un consejo aleatorio (*Ilustración 24*). Los consejos que se le muestran al usuario han sido almacenados previamente en la base de datos correspondiente. Al hacer la petición el usuario, se accederá aleatoriamente a uno de estos consejos almacenados y se le mostrará por pantalla.

Ilustración 26: Captura de pantalla prueba Petición de un consejo

BLOQUE DE FUNCIONALIDAD	FUNCIONALIDAD	PRUEBAS						
		1	2	3	4	5	6	7
TRATAMIENO DE USUARIOS	Registrarse en el sistema (1.a)	✓						
	Modificar datos personales (1.b)	✓						
	Añadir año de nacimiento (1.c)	✓						
	Añadir perfil de estudiante (1.d)		✓					
	Editar perfil de estudiante (1.e)		✓					
	Acceder al sistema (1.f)		✓					
CURSOS	Añadir un curso (2.a)			✓				
	Buscar un curso (2,b)			✓				
	Buscar sin coincidencias (2.c)			✓				
	Matricularse en un curso(2.d)				✓			
	Desapuntarse de un curso(2.e)						✓	
PLANIFICADOR ADAPTATIVO	Acceder la planificador (3.a)					✓		✓
	Marcar como realizada una tarea (3.b)					✓		
	Acceder a las tareas completadas (3.c)					✓		
TAREAS	Añadir tareas (4.a)			✓				
	Editar tareas (4.b)			✓				
	Eliminar tareas (4.c)			✓				
	Comprobar orden de tareas (4.d)			✓				
	Coger como fecha de entrega la del curso (4.e)			✓				
REALIMENTACIÓN	Almacenar tiempo de realización de la tarea (5.a)					✓		
CONSEJOS	Mostrar consejos (6.a)							✓

Tabla 11: Validación de las funcionalidades de la aplicación tal y como se recogen en el capítulo 3

4.2 Pruebas de carga

Debido a que la aplicación está diseñada para ser utilizada simultáneamente por un número elevado de alumnos, se ha tenido especial cuidado con los diferentes problemas de carga que pueda tener la aplicación. Las siguientes comprobaciones se han realizado con múltiples clientes. La aplicación se conecta a tres sistemas externos (Servidor web, *MySQL* y *MongoDB*) y por esta razón se ha comprobado qué pasaría en caso de que cualquiera de estos sistemas fallara en un momento puntual. La idea es que a pesar de que exista algún fallo momentáneo en estos sistemas, la aplicación no se bloquee ni sea necesario tener que cerrar la aplicación y volver a abrirla. No en todos los casos la aplicación se conecta a los tres sistemas externos, sino que puede conectarse a tres, a dos o a ninguno. En el caso de que no se conecte a ninguno no se harán comprobaciones, ya que se entiende que no puede existir ningún problema de carga. Para los casos en los que la aplicación se conecta a dos o tres sistemas se han realizado pruebas para comprobar qué sucedería en el caso de que fallara algún sistema.

Para los momentos en los que la aplicación se tiene que conectar exclusivamente con el servidor web y el sistema de gestión de base de datos *MySQL*, se pueden encontrar dos fuentes de error, bien que directamente el servidor web no funcione y por lo tanto la aplicación no se pueda conectar con nada externo, o bien que el servidor web sí funcione pero que al hacer las peticiones a la base de datos *MySQL* está no esté disponible.

Para los casos en los que la aplicación se tiene que conectar con el servidor web y el sistema de gestión de base de datos *MongoDB*, igual que en el caso anterior podrán existir dos fuentes de error. En este caso las fuentes de error pueden ser: fallo en el servidor o fallo en *MongoDB*. En el caso de que sea *MongoDB* el sistema en el que se produzca el fallo, pueden aparecer otros problemas. A continuación se muestra una lista de los diferentes problemas que pueden surgir si falla *MongoDB*.

- El usuario quiere obtener información de una tarea que ha sido eliminada por otro usuario.
- El usuario quiere marcar como realizada una tarea que otro usuario ha eliminado del sistema.
- El usuario quiere eliminar una tarea que ya ha sido eliminada por otro usuario.

En todos los casos se mostrará al usuario un mensaje avisándole de un problema de servidor y se mandará al usuario a la pantalla anterior para que se recarguen los datos actuales.

Para los momentos en los que la aplicación se tiene que conectar con el servidor web, el sistema de gestión de base de datos *MongoDB* y el sistema de gestión de base de datos *MySQL*, se pueden encontrar tres fuentes de error, bien que directamente el servidor web no funcione y por lo tanto la aplicación no se pueda conectar con nada externo, o bien que el servidor web y *MySQL* sí funcionen pero que al hacer las peticiones a la base de datos *MongoDB* ésta no esté disponible o bien que el servidor web funcione correctamente y *MongoDB* también pero *MySQL* no. En este caso se tiene especial cuidado ya que siempre que uno de los tres sistemas externos falle se debe de anular directamente la operación, ya que en caso contrario sería posible que se modificase la información de *MySQL* y no la de *MongoDB* o viceversa.

En el momento en el que exista un error de servidor, independientemente del sistema causante del error, al usuario se le mostrará un mensaje de alerta indicándole que ha habido un error de servidor y que lo intente más tarde. Al usuario no se le va a informar de qué tipo de error ha surgido porque se ha entendido que el usuario no necesita saber este tipo de información. En la *Ilustración 24* se muestran tres capturas de pantallas en situaciones de error de alguno de los sistemas involucrados en las distintas peticiones. En una de ellas, acceso a la aplicación, se muestra un ejemplo de conexión con el servidor y con *MySQL* (*Ilustración 24a*). En otra, se muestra un ejemplo de conexión con el servidor y con *MongoDB*, en concreto la petición del planificador (*Ilustración 24b*). En la *Ilustración 24c* se muestra el momento en el que un usuario quiere apuntarse en un curso y la aplicación tiene que establecer conexión con el servidor y con ambos sistemas de almacenamiento de información (*MySQL* y *MongoDB*).

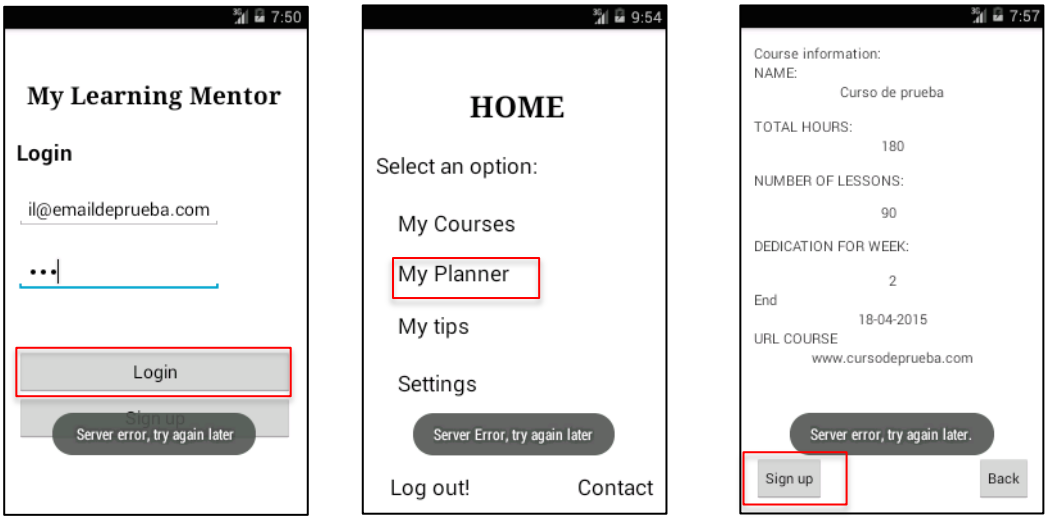


Ilustración 29: Capturas de pantalla con fallo de servidor. De izquierda a derecha: (a) situación de error en servidor o MySQL, (b) situación de error en servidor o MongoDB y (c) situación de error en servidor, MySQL o MongoDB

4.3 Seguridad de la aplicación

El único dato que contendrá seguridad adicional y se tratará de forma diferente será la contraseña que el usuario elige para poder acceder al sistema. Este campo será almacenado en la correspondiente tabla de la base de datos de *MySQL* encriptado con *SHA-1 (Secure Hashing Algorithm).*

SHA-1 es utilizado para crear un resumen que representa el mensaje. Este resumen tiene una longitud de 160 bits. En este caso, el resumen es lo único que se almacenará en la base de datos, evitando así almacenar la clave en claro para que los posible atacantes no puedan ver esta información de forma sencilla. Esta información será tratada siempre que un nuevo usuario se registre en el sistema, o bien uno registrado acceda al sistema. También puede ser tratada en el caso de que el usuario decida cambiar la contraseña [42].

Se ha elegido *SHA-1* ya que es uno de los sistemas más utilizados para guardar contraseñas encriptadas [43]. Además se ha elegido este sistema por su facilidad de uso al estar completamente integrado en *MySQL* (SGDB donde se almacenan las contraseñas).

4.4 Conclusiones

En este capítulo se han realizado diferentes pruebas para comprobar que la aplicación cumple con los requisitos y requisitos funcionales planteados en el capítulo 3, concluyéndose que todos los requisitos y requisitos funcionalidades han sido conseguidos. En este capítulo también se han realizado diferentes pruebas para comprobar que la aplicación es resistente a diferentes fallos del servidor y se ha mostrado que sucede en caso de que el servidor tenga algún problema. Por último, en este capítulo se ha decidido el sistema criptográfico a utilizar para almacenar la contraseña del usuario en el servidor siendo el elegido *SHA-1*.

Capítulo 5: Conclusiones y líneas futuras

Es este capítulo se expondrá al lector las conclusiones de la realización de este proyecto. Además se mostrarán las líneas futuras que podrá seguir la aplicación a la finalización de este proyecto.

5.1 Conclusiones

Con *MyLearningMentor* se quiere crear una aplicación que facilite a un usuario completar un curso abierto, online y masivo (MOOC). El principal problema que se quiere solucionar con *MyLearningMentor* es la gran cantidad de abandonos que tienen estos cursos, ya que solo un pequeño porcentaje de la gente que los empieza los termina. Gracias a diversos estudios se ha llegado a la conclusión de que el mayor problema al que la gente se enfrenta al realizar un curso de estas características es la falta de contacto con el profesor. En un curso presencial el profesor suele ser el encargado de marcar las pautas a seguir para el buen funcionamiento del curso. En un MOOC tiene que ser el alumno el encargado de organizar sus tareas de forma autónoma. Por tanto, *MyLearningMentor* se ofrece como ese punto de apoyo donde los usuarios pueden ordenar sus cursos, leer consejos y ver las actividades que deberían realizar con mayor prioridad. Además, *MyLearningMentor* proporciona un cierto control sobre los cursos a los que el usuario se apunta, para así indicar al usuario que no debe apuntarse a muchos cursos si no tiene tiempo suficiente para realizarlos.

MyLearningMentor ha sido implementado siguiendo una arquitectura *cliente-servidor*. El cliente ha sido implementado para que funcione con el sistema operativo *Android,* mientras que el lenguaje utilizado en la parte de servidor ha sido PHP. Como sistemas de almacenamiento de información se han utilizado dos gestores de bases de datos distintos, siendo éstos *MySQL* y *MongoDB*. La decisión de utilizar dos gestores de bases de datos se ha tomado para que la aplicación sea lo más eficiente posible. Como lenguaje de comunicación entre el cliente y el servidor se ha utilizado JSON.

Para la realización de la aplicación se ha partido de un diseño inicial proporcionado por el artículo *"Scaffolding self-learning in MOOCs"* [6]. A partir del diseño inicial se han realizado diversos cambios hasta llegar al modelo final, que es el que se ha implementado. Estos cambios han estado motivados por las tecnologías elegidas a la hora de implementar la aplicación. Sobre este modelo final se han realizado las validaciones de la aplicación comprobando que todo lo que se había planteado como posibles funcionalidades del proyecto se han implementado correctamente. También se ha comprobado que se trata de un sistema robusto frente a situaciones en las que el servidor falla. Para dicha comprobación, se han realizado algunas pruebas iniciales.

5.2 Líneas futuras.

Una vez realizada la primera versión de *MyLearningMentor* se plantean diferentes opciones por donde se podría encaminar el futuro de la aplicación. Algunas de estas ideas se muestran a continuación.

- Para una aplicación con mayor fiabilidad y estabilidad sería un punto a favor que no fueran los usuarios los que tuvieran que añadir las tareas a los cursos, sino que los usuarios solamente tengan que introducir la URL del curso y sea la propia aplicación la que se encargue de obtener todos los datos tanto del curso como de las actividades del propio curso.
- La aplicación ya recoge el tiempo que las personas tardan en realizar una tarea y lo almacena en la base de datos. La idea es que a partir de esa información, cuando la aplicación tenga una cantidad suficiente de datos, calcule una media y pueda proporcionar un valor más realista acerca de la duración de la tarea, aunque siempre utilice también la información que el profesor del curso haya estimado.
- Se podría añadir una nueva opción al menú principal donde los usuarios puedan establecer comunicaciones entre ellos para que no se sientan solos a la hora de realizar un curso y puedan comentar los puntos que les resultan más difíciles y ayudarse mutuamente.
- La aplicación tendrá no sólo en cuenta el tiempo que tiene disponible cada usuario por semana, sino también el nivel y hábitos de estudios de cada usuario. De esta forma, si el usuario está acostumbrado a estudiar será más probable que cumpla aproximadamente el tiempo que el profesor ha establecido para la tarea. Sin embargo, si el usuario no tiene hábito de estudio, es probable que necesite más tiempo para realizar la tarea. *MyLearningMentor* tendría en cuenta esto a la hora de dejar al usuario apuntarse en un curso o no.
- La aplicación no solo podría ser utilizada para cursos online, sino que también podría ser utilizada para cursos presenciales en los que el profesor quiera ofrecer a los alumnos una herramienta extra de apoyo. Para esto, se podrían crear dos perfiles distintos, el perfil “Profesor” y el perfil “Alumno”. Dependiendo del perfil con el que se acceda la aplicación, ésta permitiría o no la modificación de cursos y tareas.
- Se propone implementar la aplicación para dos sistemas operativos más iOS y Windows Phone.
- Se podrían añadir notificaciones para que no sea el usuario el que tenga que abrir la aplicación y acceder al planificador, sino que la aplicación le recuerde por medio de una notificación qué actividades debería realizar cada día.

Abstract

MOOCs are massive, open, online courses. These courses offer the possibility to access sources of quality education for all. These courses are of a great variety of themes and are offered by universities worldwide through distribution platforms of MOOCs. The most popular platforms are, for example Coursera, edX or MiríadaX. The MOOCs are a great opportunity to gain knowledge, but the main problem is that over 90% of students drop courses. For these reasons a tool is developed to help and motivate students to finish the courses.

With this purpose it is proposed to perform an application that helps students enrolled in MOOCs. The application will be a mobile application because if it is a mobile application, users can install the application in their own devices easily. The application will be performed in the Android operating system. With this decision the development tools and the distribution facilities that Android offers to the developers are exploited.

For the development of this application client-server architecture has been used and both parts will be developed. As a result, an application ready for use by any student of a MOOC is obtained. The main options offered to user try to help him to complete all his enrolled courses. The ultimate goal of the application is to help students to complete their studies. Before ending this work, several tests have been performed to check if the application meets the requirements. Finally, the conclusions that have been extracted from the development of the application will be presented.

Extended abstract

Context

Currently thousands of students have the opportunity to access free educational resources. This is possible through the MOOCs (Massive Open Online Courses). The MOOCs have had a great impact on society and have changed the education sector.

MOOCs began at the University of Stanford (Fall 2011). A teacher decided to offer a free course through internet for anyone interested. Professor got 120000 students and started a new trend in the education sector. MOOCs growth has been positive over the next three years [1].

Students can access courses through distribution platforms of MOOCs. Some of the more important distribution platforms are: *Coursera, EdX, Udacity, FutureLearn* and *MiríadaX*. For example, *Coursera* has more of 8 million students and more of 700 courses of different universities. Nowadays the platforms offer a greater number of courses with different subjects and languages [2].

Spain is the leading country in terms of MOOCs at European level (December 2013). Spain provides 35% of MOOCs that are offered in Europe. The Spanish universities are the main reason of the great amount of courses offered in Spain because a third of Spanish universities offer at least one MOOC. The more important distribution platform of MOOCs in Spain is *MiríadaX*, where more of 75% of Spanish universities have offered some MOOC [3].

One of the main reasons of the great growth of the number of MOOCs offered in Spain is the actual social-economic situation in the country, where 24% citizens are unemployed (July 2014) [4]. Many unemployed people looking at the MOOCs the opportunity to improve their skills to find a job and they decide to enroll in these courses [6].

Motivation

As MOOCs are online all people can access them and for this reason, the MOOCs have students from different countries and different economic situations. These wide varieties of people who use the courses have different profiles and different levels of knowledge.

Most of the people enrolled in a course are between 25 and 40 years old. These people often have a Bachelor's Degree, Master's Degree or PhD. However, currently there is a new common user profile; this profile is unemployed people trying to get a new job. This new profile doesn't have habits of study [6].

When performing an online course arises an inconvenience that affect of differently depending on the student profile. This problem is the lack of physical teaching staff for interacts with the student continuously because in a MOOC there is a great amount of students.

Some studies have shown that teamwork and good organization of study time are essential to successfully take an online course [6]. This is easy to achieve in a classroom course, but it can become a difficult problem when the course is online. This is understood as one of the main cause of leaving a MOOC, which only 10% of people ending [6].

As a solution to these problems arises *MyLearningMentor,* an application that allows any user to access a personalized area created from courses in which the user has been enrolled. This application will try to help student by offering them a personal support that an online course cannot offer to them.

Objectives

The work involves the implementation of an application that helps students complete the MOOCs. The objectives can be summarized in the following list:

1. Approach of the context and motivation for the realization of the application.
2. Study of the solution of the article where the application is proposed [6].
3. Study of technologies related to mobile applications and selection of the most appropriate technologies.
4. Refinement the design of the application.
5. Development of *MyLearningMentor*.
6. Validation of the proposed prototype.
7. Approach of future lines related with the application.

Initial idea of MyLearningMentor

MyLearningMentor is a proposal for architecture. This architecture has been designed from a set of requirements. *MyLearningMentor offers support to* different profiles of student in the realization of a MOOC. The offered support consists an organizational system to help students to complete with success an online course because only 10% of students completing these courses [6].

The architecture proposed in the article “Scaffolding self-learning in MOOCs” [6] has been made from requirements to be met by the final application. These requirements are summarized in *Table 1* [6].

The first requirement (Requirement 1) is that the application has to be distributed as a mobile application. This election is justified because most of the people, who use a MOOC, use a mobile phone every day [6]. The second requirement (Requirement 2) is that the application has to adapt to different student profiles. The second requirement is important because if the application doesn’t adapt to different student profiles, the application only can be used by a kind of students and in this case the application doesn’t make sense. The third requirement (Requirement 3) is that the application has to offer an adaptive planner to the different student profiles. This planner offers a list of task that the user must perform according with the courses that the user has been previously enrolled. The fourth requirement (Requirement 4) is that the application has to trust on the information entered by users.

This requirement is important because the application cannot extract all the information about tasks directly from the websites of the MOOC. The fifth requirement is that the application must to provide tips to the student for help them in the courses. These tips will be different depending of the student profiles. The last requirement (Requirement 6) is that the application can be a meeting place for students and volunteer tutors.

Identifier	Requirement
Requirement 1	Distributed as a mobile application
Requirement 2	Customizable to different students Profiles
Requirement 3	Include an adaptable daily planner
Requirement 4	Rely on crowdsourced information
Requirement 5	Provide tips and hints to make the most of MOOCs
Requirement 6	Save as a meeting point with volunteer mentors

Table 1: Requirements of the application

Architecture

MyLearningMentor is an architecture client-server for mobile devices. The choice of this kind of architecture has motived by two principal reasons: is a mobile application (Requirement 1) and the information introduced by one user is relevant for the rest of users (Requirement 4).

MyLearningMentor has four principal services. These services are: *Account & Profile Management*, *MOOC Directory*, *MOOC Activity Suggestions* and *Feedback Gathering*. The service *Account & profiles* allows to the student enroll in the system and this service also will verify the access to the application. The service *MOOC Directory* is related with MOOC's information. This service allows storing all the information of the different courses. The services *MOOC Activity Suggestions* will be responsible for providing a list of recommended tasks according to the academic level of the user. The last service *Feedback Gathering* allows completing the course information with the information entered by users.

MyLearningMentor has three databases in this first approximation: *User profile*, *MOOCs Data* and *Feedback*. The database *User Profile* will store information related to the profile of the user. The database *MOOCs Data* will store information related to the features of the courses. The last database *Feedback* will store information related to the progress of the students.

Finally, *MyLearningMentor* has two processes running periodically: *Gathering MOOC data* and *Defining activity recommendations.* The process *Gathering MOOC data* collects information from the different courses in the distribution platforms MOOCs. The process *Defining activity recommendations* selects the recommended tasks for each user.

Interface

The first thing that *MyLearningMentor* shows to the user will be a login screen, where user has to introduce an email-password pair correct for enter to the menu or to touch a button for go to registration screen.

Once user has accessed to the application can proceed to modify his personal profile and academic profile. When the user completes all his personal information will be redirected to

the screen where the user selects the MOOCs or will add new courses. With this information user can access the adaptive planner. Besides all this the user can touch an option for receive a list of tips related with study habits.

These are the main screens offered to the user. But the application will have many more screens where the user will can interact with the application.

Selection of technologies

To select the most appropriate technologies for the system have been evaluated different options. For the choice of the operating system have been evaluated: Windows Phone, BlackBerry OS, Android and iOS. Finally, the operating system used has been Android, because Android is very common in the society and has more facilities for the developers that the others operating system. To decide the server side language has been studied: PHP, ASP, JSP and PERL. The PHP language has been chosen because is fast, easy to learn and has a lot of information available. For the choice of relational database system have been evaluated *MySQL* and Postgre. *MySQL* has been selected because *MySQL* has good integration with languages of server side and it has much documentation for consult (because *MySQL* is very popular). For the choice of documental database system have been evaluated: *MongoDB*, Cassandra and CouchDB. The database system chosen has been *MongoDB* because *MongoDB* stored key-value pairs quickly and it has good integration with PHP. The studied options to choose the language of communication between client and server have been XML, YAML and JSON. JSON has been chosen because it facilitates integration with *MongoDB* and PHP.

Final design of My Learning Mentor

It has been made a refinement of the initial prototype of *MyLearningMentor.* The requirements of the application have been revised and they have been adapted for a first implementation of the application. The different requirements and functionalities used will be explained below.

Requirements

The requirements of *MyLearningMentor* application are:

- It will be a mobile application.
- It will be adapted to different user profiles.
- The application will provide the user with a customizable planner.
- The application will offer users the ability to access tips for improve his study habits.
- The application will collect information of the time the user takes to do a task.

Functionalities

The functionalities have been divided in 6 different blocks for a better understanding: users, courses, tasks, tips, adaptive planner and feedback. Each block has different functionalities to service all the options, which the application offers to the user. These functionalities are exposed in blocks below.

1. Users:
    a. The user can register in the application. The application requires the full name, email, and password. It is necessary introduce the password twice.
    b. The user can add or change his personal information.
    c. The user can add his birth year.
    d. User can add his study profile. The application requires the level of education, hours available for study and days of the week that he can study.
    e. The user can review and change student profile if he wants.
    f. Any user can access the system if he has an email-password pair correct.
2. Courses:
    a. User can add a course provided it meets two requirements. The first is that the amount of time devoted to this course per week plus the amount of required hours by his other courses is less or equal than the hours that the user has available to study in a week. The second is that URL doesn't exist in the system. The user will can add information such as the name, the total number of hours, number of lessons, hours spent per week, URL, course theme and date of start and end.
    b. Any user on the system can search for any course that is already registered in the system.
    c. The system will warn to the user if there are no results when the user makes a search.
    d. The user can view the stored information of one course and may enroll only if he meets the first requirement of (a).
    e. The user can delete is relation with a course if he wants.
3. Tasks:
    a. The user can add new tasks.
    b. The user can edit tasks.
    c. User can delete task.
    d. The system will check the order of the task in the course because it is impossible to have two tasks with the same order.
    e. If the users don't complete the information of the deadline of the task, the system will assign the date of course completion by default.
4. Tips:
    a. The user can access tips to improve his studies habits.
5. Adaptive scheduler:
    a. The user can see a planner based on his pending tasks. Tasks are sorted by a double classification (first by importance and then by deadline).
    b. Selecting one task in the planner the user can see the information of the task. The user can mark a task as completed.
    c. User can view the tasks already completed.
6. Feedback:
    a. The system stores the amount of time that the user has used to complete the task.

Final Architecture

Starting from the initial architecture it have done different modifications for adapt to the previous requirements. After performing the refinement of the initial architecture, the application will be developed.

The three databases of the initial architecture are grouped in a database. Each of the databases of the initial architecture corresponds to one table of the same database in the final architecture. These tables are: Profiles, Mooc_data and Tips. Besides, three new tables will be designed in this moment for a correct realization of *MyLearningMentor.* The first new table is an extension of the tables *Profiles and Mooc_data* and it stores the relation between user and courses. The second new table stores the tasks of the courses. The last new table stored the relation between the user, his courses and the tasks.

It has been needed to include two sets of databases, relational databases and documental databases. This is because the documental databases are more efficient when the information does not have a clear unique identifier. This happens in the task of the courses. For this reason, a new *MongoDB* database will be used for store the information of the tasks and the relation between tasks, courses and users. This information will be stored in two different collections.

In conclusion will be defined six different tables. Four of them defined as tables in a database relation and two of them as collections of document database. The databases with their tables and their collections are summarized below:

- Tables in the relational database (*MySQL*):
    - User's information.
    - Courses information.
    - Relationship between courses and users.
    - Tips.
- Collections in the documental database (*MongoDB*):
    - Tasks of the courses.
    - Relationship between user-course-task.

In relation to the services offered on the initial architecture all of them will be made but with some refinements. Besides, in the final architecture a new service will be added. The first *Account and Profile Management* is related to users account and the access to the application. The second service MOOCs directory is related to information of the courses. The third service *MOOC Activity Suggestions* is related with the adaptive planner. The fourth service *Feedback Gathering* is responsible for completing the courses information with the information introduced by the users. The last service *Tasks directory* is responsible for controlling the tasks of the different courses. The services summarized below:

- Account and Profile Management.
- MOOCs directory.
- MOOC Activity Suggestion.
- Feedback Gathering
- Tasks directory.

Implementation

Both the server side and the client side have been developed. The development of both has been developed simultaneously.

Server

On the server side it has developed necessary documents for the correct operation of the application. A total of documents 25 PHP with which the client will connect in different stages of implementation of the application. Excluding four auxiliary files, all other documents will follow a similar structure. They receive the necessary information from the client via JSON, later they establish connection to the corresponding database and sent to the client an answer also through JSON. The different documents are summarized in the *Table 2*.

Document (.php)	Description
Access	Controls access to the application.
AddCourse1	Allows a user to add a course to the system.
AddCourse2	Enrolls to a user in a course.
AddUser	Adds a new user in the system.
CourseInformation	Returns the information of a course.
PersonalInformation	Returns the information of a user and adds or updates personal information.
StudiesInformation	Returns information of academic profile of a user.
AddStudiesInformation	Adds or modifies academic profile information of a user.
Tips	Returns a random tip.
QuitCourse	Deletes the relation between a course and a user.
AllMyCourse	Returns the courses of a user.
UpdateStateTask	Updates the state of a task.
DeleteTask	Deletes a task of a course.
TaskInformation	Returns the information required for a task.
AddTask	Adds a new task in a course.
MyTaskComplete	Returns the completed tasks of the user.
MyTaskPending	Returns the pending tasks of the user.
TaskInformation2	Returns information of a task.
UpdatePersonalSetting	Updates or adds personal information of a user.
BdConnect	Establishes connections with *MySQL* database.
ConfigBD	Contains the necessary information of *MySQL* database.
FunctionsBD	Contains functions relate to the *MySQL* database.
FunctionsBD2	Contains more functions relate to the *MySQL* database.

Table 2: PHP documents

Client

The client has been developed for Android operating system. It has been created a new Android project, which has two packages. One of these has all activities of the new application with their respective parts as classes or layout. The other package has been used to facilitate the Android client connections with the server. This package already existed and it has been used in this application [38]. The client will be connecting with the server always that he requires connect with a database. All activities of the package that defines the application are summarized below. The different activities are summarized in the *Table 14*.

Name	Description
Login	Screen to access the application.
AddUser	Enter data for a new user.
Setting	Options settings.
Home	Main screen where the user chooses to do.
Personal	Modify the personal details or add the year of birth.
Tips	Displays a random tip.
Studies	Adds studio settings.
MyPlanner	Displays the adaptive planner.
View Studies	Displays the study settings already stored in the system.
SeeTask	Displays the information of the task and the user can save the task as done.
Complete	Already tasks completed by the user are displayed.
SeeTask2	Displays the task information.
SeeTask3	Displays the task information from the completed tasks.
AddTask	Used to add a new task in a course.
SearchCourse	Screen to perform a search.
InforOtherCourse	Displays course information that is accessed from the browser.
Courses	Displays the list of courses in which the user is registered.
OtherCourses	Displays a list of search results.
Task	Displays the tasks of a course.
AddCourse	Adds information on a new course.

Table 3: Activities of the application

Connections between client and server

The server has to connect with the both databases systems used (*MySQL* and *MongoDB*). Depending of the request of the user, the application will establish connection with some of the two systems or both systems at the same time. The possible connections are summarized in the *Table 15*.

File type	File name (.php)
Connection *MySQL*	Access AddCourse1 AddCourse2 AddUser CourseInformation PersonalInformation StudiesInformation AddStudiesInformation Tips AllMyCourses
Connection *MongoDB*	UpdateStateTask DeleteTask TaskInformation AddTask MyTaskComplete MyTaskPending TaskInformation2
Connection *MySQL* y *MongoDB*	UpdatePersonalSetting AddCourse1 AddCourse2 QuitCourse

Table 4: Relation between client and server

Validation

To verify that the application provides the requirements and functionalities have been made several tests (a total of seven tests). The first test *Registration* includes actions to check of a part of functionalities of the *User* block. The first test simulates that a new user registers into the application and the user updates their personal settings. The second test *Update academic profile* checks the rest of the functionalities of user block. In this test a user decides update his academic profile for this the user enters to the study preferences from the menu of settings. For check the second (*Courses*) and third requirements blocks (*Tasks*) two tests have been performed: *Add a new course* and *Enroll in a course.* The third test *Add a new course* consists in that a user decides add a course that isn't in the system. The user introduces all the information of the course in the third test. Besides in this test the user modify the tasks of the course. The fourth test *Enroll in a course* checks that a user can sign up in an existing course. The user can enroll in a course only if he has available time. For check the functionalities of the block *Adaptive planner* have been performed two new tests: *Management adaptive planner* and *Update the adaptive planner.* In the test *Management adaptive planner* is checked that a user can access to a personal adaptive planner from main menu. In the test *Update the adaptive planner* the user decides delete his relation with a course and then he access to his adaptive planner and he checks that in his planner aren't the tasks of the deleted course. The last test performed checks that the user can access to a random tip when he wants from the principal menu of the application.

All the tests and the functionalities of the application are related with the tests in the *table 16.* Each functionality has a ✓ in the test where it was checked.

BLOCK OF FUNCTIONALITIES	FUNCTIONALITY	TESTS						
		1	2	3	4	5	6	7
USERS	Registration in the system (1.a)	✓						
	Modify personal setting (1.b)	✓						
	Add new of birth (1.c)	✓						
	Add study profile (1.d)		✓					
	Edit study profile (1.e)		✓					
	Log in the system (1.f)		✓					
COURSES	Add a course (2.a)			✓				
	Search a course (2,b)			✓				
	Search without results (2.c)			✓				
	Enroll in a course (2.d)				✓			
	Delete the relation with a course (2.e)						✓	
ADAPTIVE PLANNER	Access the adaptive planner (3.a)					✓		✓
	Mark a task as completed (3.b)					✓		
	Access to the completed tasks (3.c)					✓		
TASKS	Add tasks (4.a)			✓				
	Edit tasks (4.b)			✓				
	Delete tasks (4.c)			✓				
	Check the order of tasks (4.d)			✓				
	Deadline of the courses (4.e)			✓				
FEEDBACK	Save the time used for complete a task (5.a)					✓		
TIPS	Offer to tips (6.a)							✓

Table 5: Relation between functionalities and tests

Loading problems

Because system has been designed to be use by a large number of students, the application has careful with the different problems that the server can have. Some test has been performed to check the result of a server failure. If there is a problem in the server, the application should not stop working for avoid that a user has to close the application.

The application doesn't connect always with all external system. When the user makes a request to the server the application can be connected to two or three external systems. The application can connect to the server + *MySQL* or server + *MongoDB* or server + *MySQL* + *MongoDB*. For these cases some tests have been done. When the application connects with *MongoDB* system some problems can appear because is possible that another user has modified this information a moment before.

- The user wants to get information from a task that another user has deleted.
- The user wants to mark a task as done that another user has deleted.
- The user wants to delete a task that another user has deleted.

The application takes special care when the user needs to modify the two systems different databases (*MySQL and MongoDB*) because the application can't permit that in these cases only a database system has been update. If a system has a problem and can't complete the request, the application must cancel the entire action.

In all cases the application has the same behavior for the user, the application displays an error message indicating that an error has occurred on the server. The server problem is solved in this way because the application understands that the user doesn't want to know that system has specifically failed.

Security application

The only data that will contain additional security will be the user password. The password is stored encrypted in the database with *SHA-1* (Secure Hashing Algorithm). *SHA-1* is used to do an abstract the message. This message has a length of 160 bits and it will be stored in the database instead of the password [42].

SHA-1 has been choice because is one of the most system more used for encrypted passwords. Also *SHA-1* has been choice because *SHA-1* has a great adoption in *MySQL* system (database system used for store the password).

The server will handle this encrypted information when a new user decides to sign up in the system or a registered user decides login in the application. In addition the server will need to modify the encrypted information when a user decides to change his password from the personal settings menu.

Conclusion

MyLearningMentor architecture presents architecture for a mobile application. This mobile application aims to help to the students of MOOCs to complete these courses with success.

The main problem to be solved by *MyLearningMentor* is the large dropouts who have these courses, because only a small percentage of the people, who started a course, get finish this course.

MyLearningMentor has been implemented following client-server architecture. Client has been implemented to run on the Android operating system, while the language of the server chosen has been PHP. As storage system has been used two engines different of databases, *MySQL* and *MongoDB*, each used at certain times. As a language of communication between the client and server it has been used JSON.

For the realization of the application, it has started from an initial design provided by article "Scaffolding self-learning in MOOCs" [6]. From the initial prototype there have been performed several changes and has been obtained a final version which is what has been implemented. Several tests have been performed for checking that functionalities of the project have been achieved. It has also been proven that the application is a robust system against possible server failures.

Future lines

One time implemented the first version of the application possible future lines of the application will be presented below.

- For an application with more reliability and stability would be a plus that users would not have to add the tasks of the courses. When a user adds a URL of a course *MyLearningMentor* would be the responsible for taking the information of the relevant pages.
- The application obtains the information that the user uses to complete a task. The new idea is that the application performs an average with this information and the application displays this average for all other user. Also is important that the user can see always the time estimated by the teacher. The idea is that the application displays to the students the two estimates.
- Add a new section where users can establish communication with other students and discuss the most difficult tasks. This can be a significant improvement because so students can feel supported and share concerns from their own place of study.
- The application now controls the number of courses that a user can enroll with the number of hours available per week. The new idea is that the application also controls the level of studies of the student and not just the hours available to calculate the number of courses that permits enroll to the student.
- The application could be not only for online courses, also could be used for classroom courses where the professor wants to give students an extra support tool. For this it could create two different profiles (*Professor* and *Student*). The user, depending of kind of profile, may or not modify the information of the courses and tasks.
- The application could be implemented to some operating system more (for example, iOS and Windows Phone).

- The application could be send notifications to the user reminding the tasks that he has to do. Of this form the application facilitates to the user the information without the user has to open the application.

Anexo A: Planificación

En la *tabla 12* se muestra un resumen de todas las actividades que se han necesitado llevar a cabo para la realización del proyecto. Además se muestra la fecha de inicio de cada tarea y la duración de cada actividad.

Actividad	Fecha de inicio	Duración (días)
Documentación del proyecto	25/01/2014	3
Estudio de las tecnologías a utilizar	28/01/2014	3
Configuración del entorno de desarrollo	31/01/2014	2
Refinamiento del diseño	02/02/2014	8
Programación cliente (Android)	10/02/2014	150
Programación servidor (PHP + Bases de datos)	10/02/2014	150
Pruebas	10/07/2014	8
Escritura de la memoria	18/07/2014	62
DURACIÓN TOTAL		386

Tabla 12: Resumen de las actividades con sus fechas de inicio y duración

A partir de las tareas expuestas en la *tabla 12,* se muestra en la *Ilustración 30* un diagrama de Gantt mostrando la evolución temporal de cada tarea. Todas las tareas son tareas críticas, ya que el retraso en alguna de ellas supondría una demora en el tiempo de finalización del trabajo.

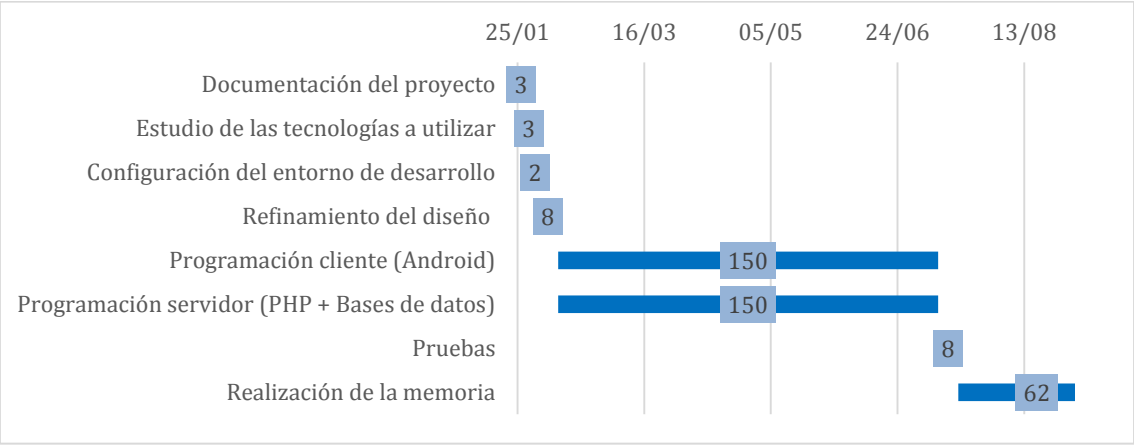


Ilustración 30: Diagrama de Gantt

Anexo B: Presupuesto

En la *tabla 13* se muestra un resumen de todas las horas dedicadas al proyecto. A partir de la cantidad total de horas utilizadas para la realización del proyecto se estima el gasto total para cubrir las necesidades de personal. El coste de hora por persona se ha extraído según los datos del coste mensual de contratación laboral de la Universidad Carlos III de Madrid en el año 2014. Para obtener esta información se ha consultado en la OTRI (Oficina de Transferencia de los Resultados de la Investigación). En la realización del proyecto han participado un titulado medio a media jornada (23.5€/horas) y dos titulados doctores al 10% de la jornada (30€/hora). Por lo tanto, según lo expuesto en la *tabla 14* el gasto de personal asciende a 33.512€.

ACTIVIDAD	DEFINICIÓN	DÍAS DEDICADOS	HORAS DEDICADAS TITULADO MEDIO	HORAS DEDICADAS TITULADO DOCTOR
1	Documentación del proyecto	3	12	2,4
2	Estudio de las tecnologías a utilizar	3	12	2,4
3	Configuración del entorno de desarrollo	2	8	1,6
4	Refinamiento del proyecto	8	32	6,4
5	Programación cliente y servidor	150	600	120
6	Pruebas y validación de la aplicación	8	32	6,4
7	Realización de la memoria del proyecto.	62	248	49,6
TOTAL DE DURACIÓN		236	944	188,8

Tabla 13: Resumen de las actividades con el tiempo dedicado a cada una de ellas

PRECIO POR HORA (€)	TOTAL DE HORAS	PRECIO TOTAL PERSONAL(€)
23,5	944	22.184
30	188,8	5.664
30	188,8	5.664
COSTE TOTAL DEL PERSONAL		33.512

Tabla 14: Presupuesto del personal

En la *tabla 15* se muestra un resumen del presupuesto necesario para hacer frente a los costes del material necesario. Para establecer este coste se ha tenido en cuenta que un ordenador de gama media actualmente puede ser utilizado aproximadamente durante cinco años. Para establecer el coste del local se tiene en cuenta que la duración aproximada del proyecto es de ocho meses. El coste total del material asciende a 1.400 €.

MATERIAL	PRECIO	COSTE(€)
Ordenador personal gama media	1000 €	200
Local	150 €/MES	1.200
COSTE TOTAL DEL MATERIAL		1.400

Tabla 15: Coste del material

En la *tabla 16* se muestra el total del presupuesto que asciende a 42.243,52€.

COSTES	**PRECIOS (€)**
PERSONAL	33.512
MATERIAL	1.400
SUBTOTAL	34.912
I.V.A. (21%)	7.331,52
TOTAL	42.243,52

Tabla 16: Presupuesto total

Anexo C: Marco legal

Relacionado con las aplicaciones móviles pueden surgir problemas legales, como la posible violación de privacidad de los usuarios. Por eso tanto a nivel nacional como a nivel europeo estos problemas han sido tratados por los organismos correspondientes.

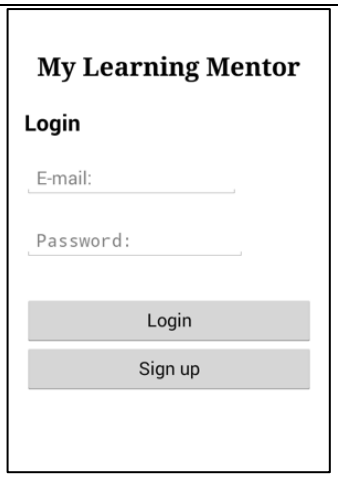
A nivel europeo cabe destacar el dictamen conjunto sobre la privacidad en las aplicaciones móviles de marzo de 2013, que se refiere a la necesidad de la protección de los derechos de los usuarios. Esta ley no sólo hace referencia a los creadores, sino que también afecta a todas las personas involucradas en el proceso, incluido el campo de la distribución. Se puede destacar como punto clave del dictamen la necesidad de consentimiento por parte del usuario cuando se le informa de los datos que se van a tratar. Además con carácter general se reconoce a todos los usuarios el derecho de acceso, rectificación y cancelación de los datos entregados. También cabe destacar lo dispuesto en el dictamen sobre la responsabilidad de las distintas personas que intervienen a lo largo del proceso de las aplicaciones móviles, estableciendo cuál es la responsabilidad de cada una de ellas respecto a los usuarios. En el caso de los desarrolladores deben de tener en cuenta que aunque su finalidad es proponer nuevo servicios a los usuarios, los nuevos servicios pueden conllevar serios problemas si no se trata con cuidado cierta información personal [44].

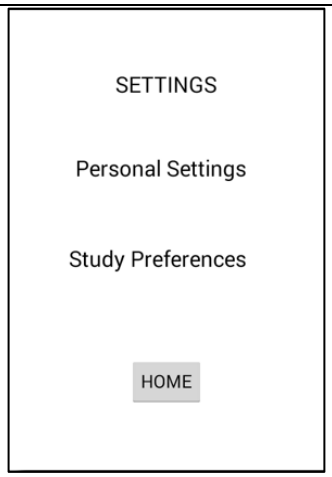
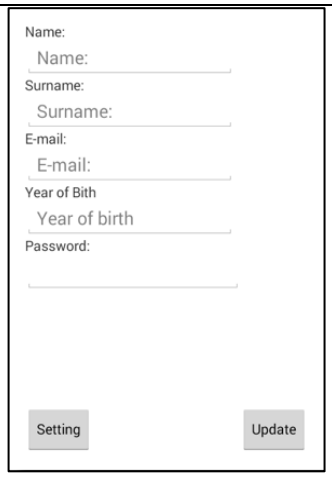
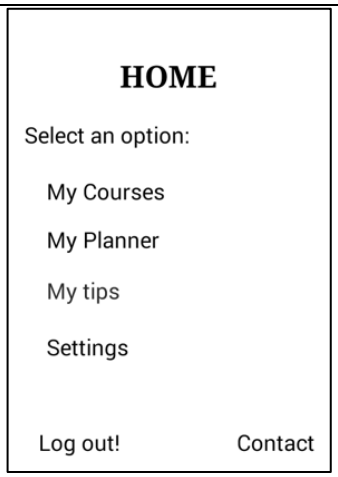
Junto a este dictamen, también a nivel europeo, se debe considerar la Directiva de Datos 95/46, así como la Directiva 2002/58/CE de Privacidad y Comunicaciones Electrónicas. En éstas se destaca la necesidad de obtener el consentimiento del usuario previamente [44].

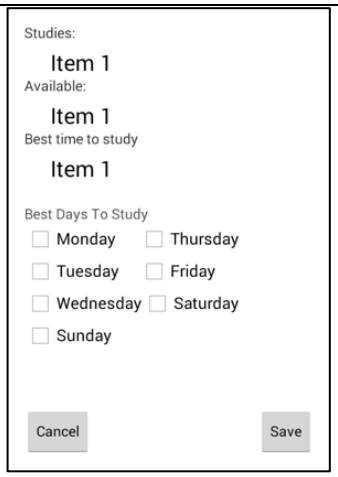
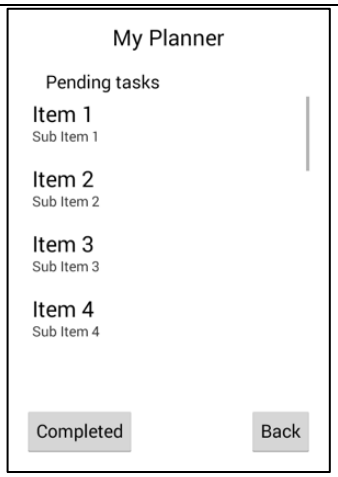
En el ámbito nacional, relacionado con la protección de datos y la privacidad, destacan la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico y la Ley Orgánica de Protección de Datos de Carácter Personal. En la primera se indica la necesidad de establecer un informe con la información personal del responsable de la aplicación. Además, en esta ley se regula el tema de publicidad en las aplicaciones. En la Ley Orgánica de Protección de Datos de Carácter Personal se hacen referencia a muchos derechos del ciudadano entre los que figuran el derecho de acceso, rectificación, cancelación e indemnización [45][46][47].

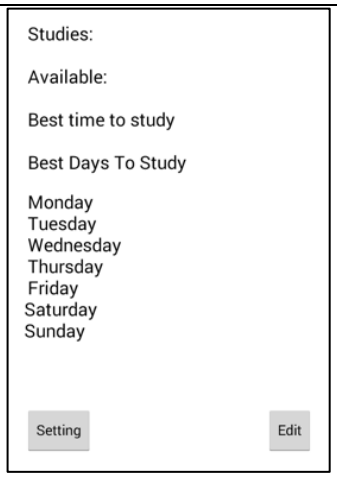
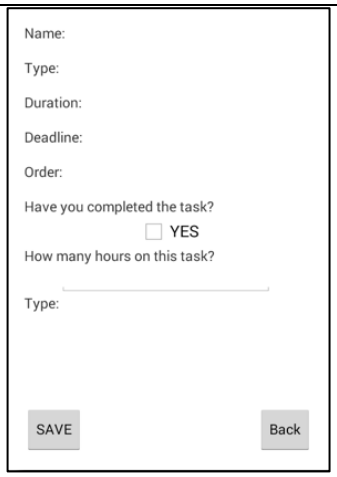
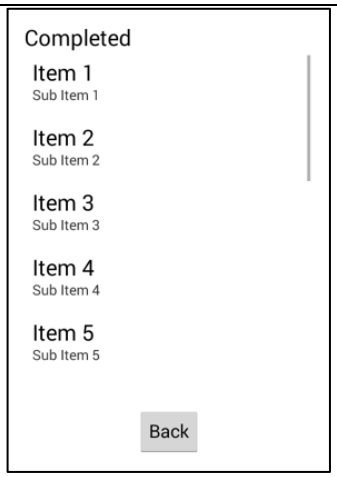
Anexo D: Actividades del cliente

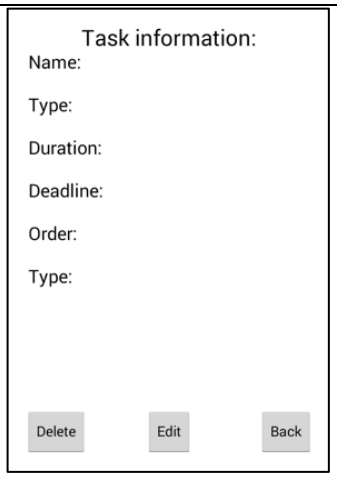
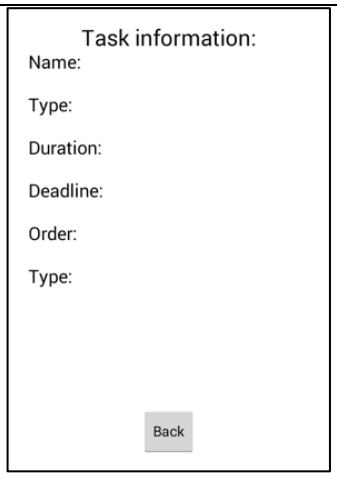
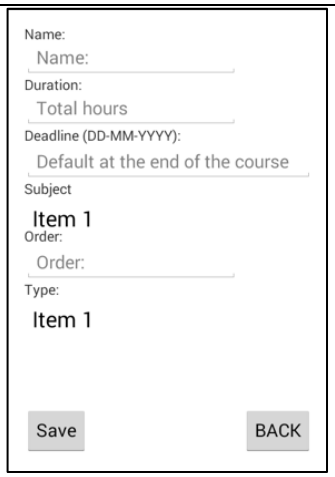
En la *tabla 22* se muestran todas las actividades existentes en la aplicación. Se expondrá una pequeña descripción de la actividad, los recursos del servidor con los que se puede conectar y una captura de pantalla de la actividad.

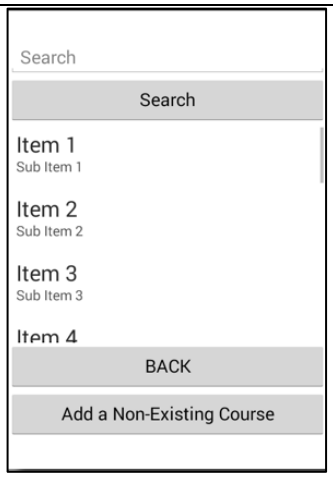
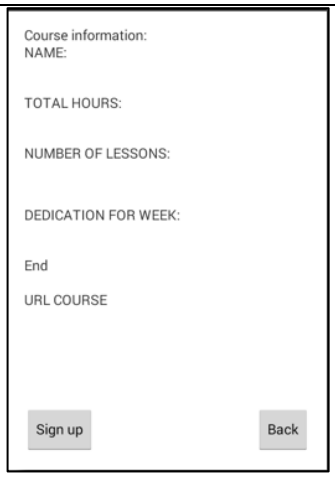
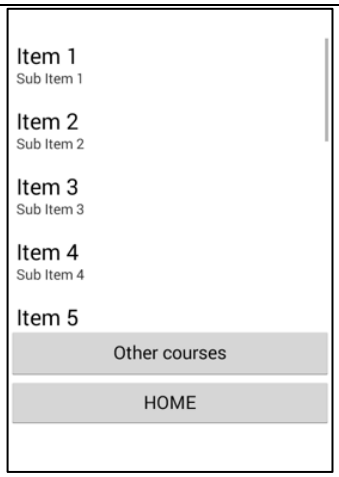
Nombre de la actividad y explicación	**Recursos del servidor(.php)**	**Captura de Pantalla**
Login Pantalla para acceder a la aplicación.	Access	
AddUser Introducir los datos para un nuevo usuario.	AddUser	

<u>Setting</u> Opciones de los ajustes.	PersonalInformation StudiesInformation	
<u>Personal</u> Modificar los datos personales o añadir el año de nacimiento.	UpdatePersonalInformation	
<u>Home</u> Pantalla principal donde el usuario elige qué hacer.	Tips MyTaskPending AllMyCourses	

Tips Muestra un consejo de forma aleatoria.		
Studies Permite añadir los ajustes de estudio.	AddStudiesInformation	
MyPlanner Muestra el planificador Adaptativo.	MyTaskComplete TaskInformation	

View Studies Muestra los ajustes de estudios ya guardados en el sistema.	AddStudiesInformation	
SeeTask Muestra la información de la tarea y marcar la tarea como realizada.	MyTaskPending UpdateStateTask	
Complete Se muestran las tareas ya completadas por el usuario.	MyTaskPending TaskInformation	

SeeTask2 Muestra la información de la tarea cuando accede desde el curso.	DeleteTask TaskInformation2	
SeeTask3 Muestra la información de la tarea desde las tareas completadas.	TaskInformation MyTaskComplete	
AddTask Sirve para añadir una nueva tarea en un curso.	TaskInformation AddTask	

SearchCourse Permite poner una palabra para realizar una búsqueda de los cursos.	AllCourses	
InforOtherCourse Muestra la información de un curso al que se ha accedido desde el buscador.	CourseInformation AddCourse2	
Courses Muestra la lista de los cursos en los que el usuario está registrado.	CourseInformation	

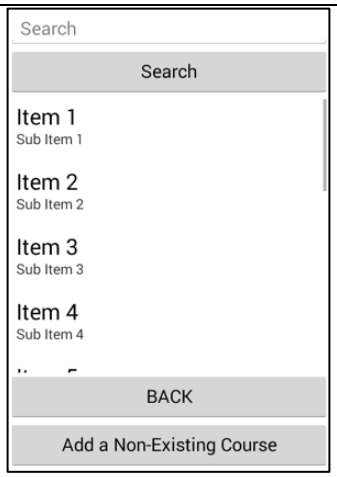
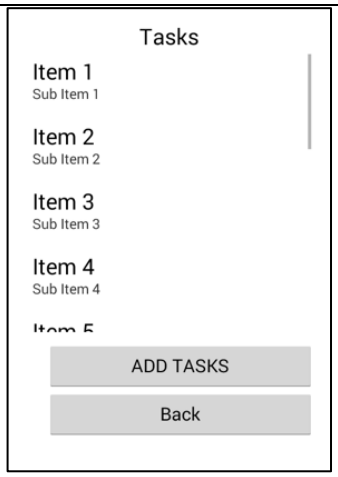
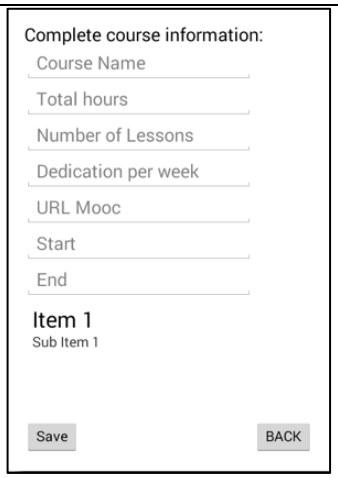
OtherCourses Muestra una lista con los resultados de una búsqueda.	AllCourses	
Task Se muestran las tareas de un curso.	CourseInformation TaskInformation	
AddCourse Permite añadir la información de un nuevo curso.	AddCourse	

Tabla 17: Actividades del cliente

Bibliografía

[1] Merche Bort. Formación gratuita y de calidad ¿Aún no has probado los MOOC? URL: http://www.emagister.com/blog/principales-plataformas-mooc/ (fecha de consulta: 26/07/2014)

[2] Coursera. URL: https://www.coursera.org (fecha de consulta: 26/07/2014)

[3] Miriada X. Informe MOOCs en España de la catedra telefónica-UPF. URL: https://www.miriadax.net/blog/-/blogs/informe-moocs-en-espana-de-la-catedra-telefonica-upf (fecha de consulta: 08/09/2014)

[4] Diana Gutierrez. Los MOOCs en España, un nuevo tipo de formación online en auge. URL: http://www.seas.es/blog/e-learning/moocs-en-espana-lideres-en-formacion-online/ (fecha de consulta: 08/09/2014)

[5] Instituto Nacional de Estadística. Encuesta de población activa. URL: http://www.ine.es/inebaseDYN/epa30308/epa_inicio.htm (fecha de consulta: 08/09/2014)

[6] Gutiérrez-Rojas, I., Alario-Hoyos, C., Pérez-Sanagustín, M., Leony, D., Delgado-Kloos, C., Scaffolding self-learning in MOOCs, Proceedings of the 2nd MOOC European Stakeholders Summit, EMOOCs 2014, 43-49, 2014.

[7] Karen Melissa Rojas Lizarazo, Jaime Esteban Roa Castañeda, Andrea Catherine Alarcón Aldana. Desarrollo de aplicaciones móviles bajo la plataforma de IPhone. Vol. 20, Nº. 31, páginas 77-91, 2011.

[8] Antonio Ortiz. El techo del iPad/el techo de los tablets. URL: http://www.error500.net/el-techo-de-ipad-el-techo-de-los-tablets/ (fecha de consulta 07/09/2014)

[9] Hugo Liria. iOs gana Mercado en España pero sigue lejos de Android. URL: http://www.kantarworldpanel.com/es/Noticias/iOS-gana-mercado-en-Espaa-pero-sigue-lejos-de-Android (fecha de consulta: 07/09/2014)

[10] Windows Phone. URL: http://www.elotrolado.net/wiki/Windows_Phone (fecha de consulta: 07/07/2014)

[11] Yare Saavedra. La evolución de BlackBerry a través del tiempo. URL: http://alt1040.com/2013/09/evolucion-blackberry (fecha de consulta: 07/07/2014)

[12] Alberto Cifuentes. BlackBerry: la evolución de su sistema operativo en imágenes. URL: http://articulos.softonic.com/evolucion-sistema-operativo-blackberry-imagenes (fecha de consulta 07/07/2014)

[13] Estas son las principales características de BlackBerry 10. URL: http://www.europapress.es/portaltic/software/noticia-son-principales-caracteristicas-blackberry-10-20130130175653.html (fecha de consulta: 07/07/2014)

[14] RW. La historia del sistema iOS de Apple, en imágenes. URL: http://www.reasonwhy.es/actualidad/tecnologia/la-historia-del-ios-de-apple-en-imagenes (fecha de consulta: 07/07/2014)

[15] Desarrollador Android Vs Desarrollador iOS. URL: http://www.ideup.com/blog/desarrollador-android-vs-desarrollador-ios (fecha de consulta: 09/09/2014)

[16] Carla L. G. Hurtado. La evolución de iOS. URL: http://www.parentesis.com/noticias/software_aplicaciones/La_evolucion_de_iOS (fecha de consulta: 07/07/2014)
[17] IOS 6 vs IOS 7. URL: http://wookeo.com/2013/actualidad/ios-6-vs-ios-7 (fecha de consulta 07/07/2014)
[18] Manuel López Michelone. La historia de Android. URL: http://www.unocero.com/2013/09/23/la-historia-de-android/ (fecha de consulta: 07/07/2014)
[19] Juan Francisco. Historia de Android: La evolución a lo largo de sus versiones. URL: http://androidzone.org/2013/05/historia-de-android-la-evolucion-a-lo-largo-de-sus-versiones/ (fecha de consulta: 07/07/2014)
[20] Juan Pablo Bustos Thames. Ingeniería en Sistemas de Información. URL: http://es.slideshare.net/jpbthames/arquitectura-de-sistemas-distribuidos (fecha de consulta: 08/07/2014)
[21] Fahmi P. Rahman. Connection between PHP (Server) and Android (client) Using HTTP and JSON. URL: http://fahmirahman.wordpress.com/2011/04/21/connection-between-php-server-and-android-client-using-http-and-json/ (fecha de consulta: 08/07/2014)
[22] Manuel Sierra. Qué es un servidor y cuáles son los principales tipos de servidores (proxy, dns, web, ftp, smtp…). URL: http://aprenderaprogramar.es/index.php?option=com_content&view=article&id=542:que-es-un-servidor-y-cuales-son-los-principales-tipos-de-servidores-proxydns-webftpsmtp&catid=57:herramientas-informaticas&Itemid=179 (fecha de consulta: 15/07/2014)
[23] Diferencias entre Software Libre y Software Comercial. URL: http://www.informatica-hoy.com.ar/aprender-informatica/Diferencias-entre-Software-Libre-y-Software-Comercial.php (fecha de consulta: 08/07/2014)
[24] Adrián Martínez Orozco. Tipos de gestores de bases de datos. URL: http://gestoresadrian.blogspot.com.es/ (fecha de consulta: 08/07/2014)
[25] Leydi Alvarez. Bases de datos. URL: http://basededatosutili.blogspot.com.es/2012/10/base-de-datos.html (fecha de consulta: 08/07/2014)
[26] Oracle, *MySQL*, SQLserver, PostgreSQL. URL: http://tustrabajosdeasir.files.wordpress.com/2013/05/gestoresbbdd.jpg (fecha de consulta: 09/07/2014)
[27] *MySQL*. 1.4 Panorámica del sistema de gestión de bases de datos *MySQL*. URL: http://dev.mysql.com/doc/refman/5.0/es/what-is.html (fecha de consulta: 09/07/2014)
[28] Hugo Rincón. BD Relacionales vs BD documentales. URL: http://es.slideshare.net/refreshmcbo/db-relacionales-vs-db-documentales (fecha de consulta: 09/07/2014)
[29] Dr. Diego Lz. de Ipiña Glz. De Artaza. Base de datos no relacionales (No SQL) URL: http://es.slideshare.net/dipina/nosql-cassandra-couchdb-mongodb-y-neo4j (fecha de consulta 09/07/2014)

[30] Office. Operaciones LEFT JOIN, RIGHT JOIN. URL: http://office.microsoft.com/es-es/access-help/operaciones-left-join-right-join-HA001231489.aspx (fecha de consulta: 08/09/2014)
[31] SQL AUTO INCREMENT Field. URL: http://www.w3schools.com/sql/sql_autoincrement.asp (fecha de consulta: 16/09/2014)
[32] PHP. La clase *MongoDB*. URL: http://php.net/manual/es/class.mongodb.php (fecha de consulta: 15/07/2014)
[33] Lenguaje De Programación Del Lado Servidor. URL: https://docs.google.com/presentation/d/1ZQgIsW4KCyIfH8NyiqH9ueanM83Ym7bjitaxyQKS0hs/edit?pli=1#slide=id.i0 (fecha de consulta: 09/08/2014)
[34] Juan Manuel Barrios. ¿Qué son los servlets? URL: http://users.dcc.uchile.cl/~jbarrios/servlets/general.html (fecha de consulta 16/09/2014)
[35] Introduction XML. URL: http://www.w3schools.com/xml/xml_whatis.asp (fecha de consulta: 09/07/2014)
[36] YAML Ain't Markup Language URL: http://www.yaml.org/ (fecha de consulta 09/07/2014)
[37] Introducing JSON URL: http://json.org/ (fecha de consulta 09/07/2014)
[38] Sebastian Cipolat. Login en Android usando PHP y *MySQL*. URL: http://androideity.com/2012/07/05/login-en-android-usando-php-y-mysql/ (fecha de consulta: 09/07/2014)
[39] UML. URL: www.uml.org (fecha de consulta: 16/09/2014)
[40] The ObjectAid UML Explorer. URL: www.objectaid.com (fecha de consulta: 16/09/2014)
[41] Cacoo. URL: http://cacoo.com (fecha de consulta: 19/09/2014)
[42] Secure Hashing Algorithm (SHA-1) URL: http://www.packetizer.com/security/sha1/ (fecha de consulta: 25/08/2014)
[43] Tesina. Universidad Tecnológica Israel. Jenny Marítza León Sangurima. URL: http://186.42.96.211:8080/jspui/bitstream/123456789/501/1/tesis%20completa.pdf (fecha de consulta: 16/09/2014)
[44] ABC. Se aprueba el primer dictamen en de privacidad en las apps móviles. URL: http://www.abc.es/tecnologia/moviles-aplicaciones/20130315/abci-aprueba-dictamen-privacidad-201303151817.html (fecha de consulta: 31/08/2014)
[45] Cristina Sandoval. José Carlos Moratilla. Privacidad y seguridad en las aplicaciones móviles. URL: http://www.audea.com/privacidad-y-seguridad-en-las-aplicaciones-moviles/ (fecha de consulta: 31/08/2014)
[46] Base de datos de Legislación. Ley orgánica 15/1999, de 13 de diciembre de protección de datos de carácter personal. URL: http://noticias.juridicas.com/base_datos/Admin/lo15-1999.t1.html#t1 (fecha de consulta: 31/08/2014)
[47] Bases de datos de legislación. Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico. URL: http://noticias.juridicas.com/base_datos/Admin/l34-2002.t3.html (fecha de consulta: 31/08/2014)